Aiden Pratt

Assignment 3 analysis document

Like most of our assignments, a big part of the project is doing algorithmic analysis and performing timing experiments.

Answer the following questions in a format that's easy for TAs to read (If you want to submit something other than PDF, ask first) and commit it to your git repo.

1. If you had backed the sorted set with a Java List instead of a basic array, summarize the main points in which your implementation would have differed. Do you expect that using a Java List would have more or less efficient and why? (Consider efficiency both in running time and in program development time.)

the arraylist or Java List dynamically grows itself as the size of the list is increased. My set with basic arrays is built with a fixed starter capacity and then grown when necessary with a growArray method.
The Java List would be more efficient in removing and adding to the array as it automatically resizes. In the basic array I made, the elements need to be shifted frequently when adding or removing.
They both would be similar in efficiency time when it comes to searching within the array.
For development time, it takes much longer to create this basic array which implements a sorted set than creating an array from a Java List.

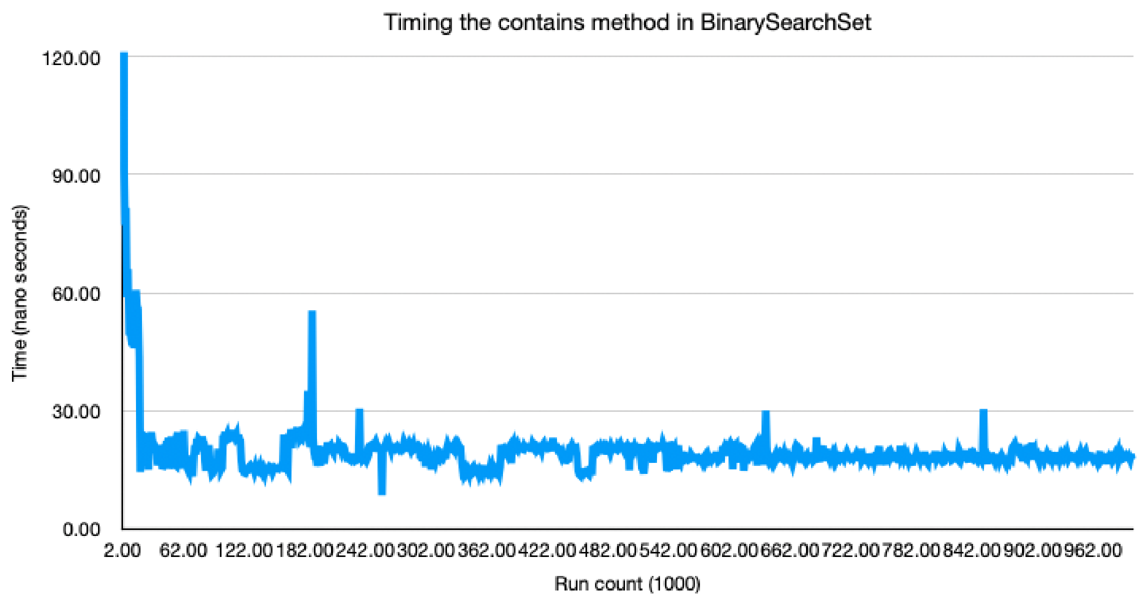
2. What do you expect the Big-O behavior of BinarySearchSet's contains method to be and why?

   O(N)

   the big-O behavior should be O(N) because the method will need to loop through the array once to find the element it is searching for. It is possible that the element it is searching for appears earlier so the method may not have to go through the loop N times. And if the element is not in the array it will still be O(N).

3. Plot the running time of BinarySearchSet's contains method, using the timing techniques demonstrated in previous labs. Be sure to use a decent iteration count to get a reasonable average of running times. Include your plot in your analysis document. Does the growth rate of these running times match the Big-oh behavior you predicted in question 2?

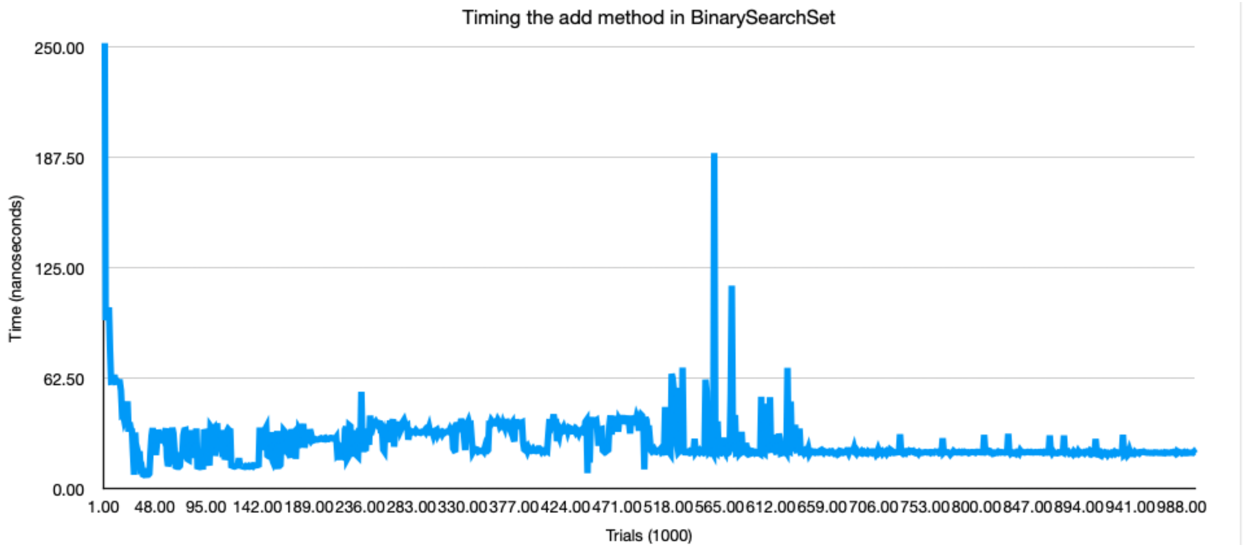
Yes because the growth rate is not getting larger as the trial is repeated. the trial is fairly consistently taking the same amount of time to run through the array and find the desired element.



4. Consider your add method. For an element not already contained in the set, how long does it take to locate the correct position at which to insert the element? Create a plot of running times. Pay close attention to the problem size for which you are collecting running times. Beware that if you simply add N items, the size of the sorted set is always changing. A good strategy is to fill a sorted set with N items and time how long it takes to add one additional item. To do this repeatedly (i.e., iteration count), remove the item and add it again, being careful not to include the time required to call remove() in your total. In the worst-case, how much time does it take to locate the position to add an element (give your answer using Big-oh)?

In the worst case, to add an element will involve having to grow the capacity of the array as well. which will involve several steps such as binary search, setting insertion point, and growing the array. This is O(N). when doing the general binary search to find the insertion point it would be O(logN) in the worst case. Then assign the element to the insertion point. the add method is dominated by the shifting steps. when growing the array, it is also just O(N) complexity because it is linear with the size and number of elements in the array. another time consuming part is copying the elements to a new array.

In this example, the first trial takes 500 nanoseconds. The rest are closer together and average 20 nanoseconds. However in the middle around trial 500, there are some outliers that are averaging 190 nanoseconds. I suspect this is due to the need to grow the array.


Timing the add method in BinarySearchSet
Time (nanoseconds)
Trials (1000)