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.)**

Running Time Efficiency:

Array: Generally faster for indexed access and has a smaller memory footprint.

List: May introduce some overhead due to dynamic resizing and object wrappers for primitives.

Development Time:

Array: Longer development time due to manual management of resizing and element repositioning.

List: Shorter development time owing to built-in methods handling most of the complexities.

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

Expect:O(logN), because I use binary search, each time I find specific position in half of current size.

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?

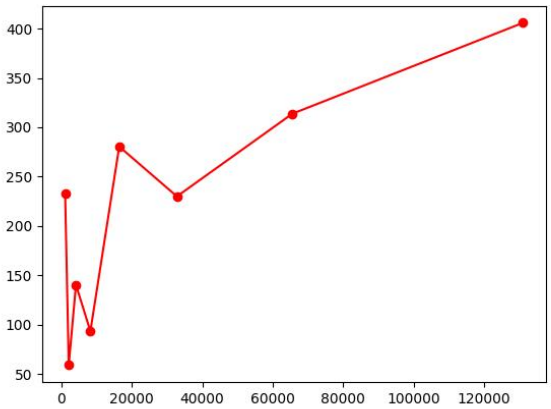

Table 1 contains method for binnarySearchSet

Yes, almost like O(logN)

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 my add method, since I use binary search, it should take around O(logN)

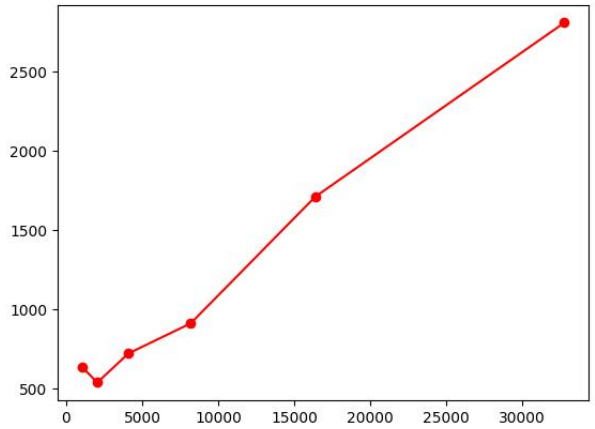

Table 2 add method for smaller size of array

With observation (Table 2), when the size of array not that large, plot show the trend of adding a new element approach to O(logN)

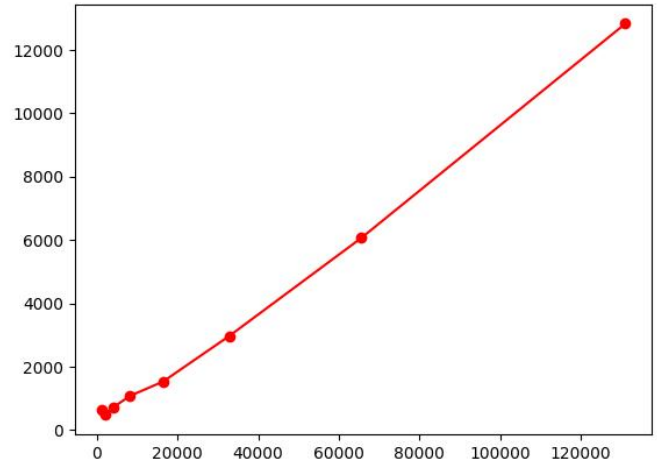

Table 3 add method for larger size array

But when the size become larger, it close to O(N)(Table 3)

In worst case, it takes O(N)