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.)

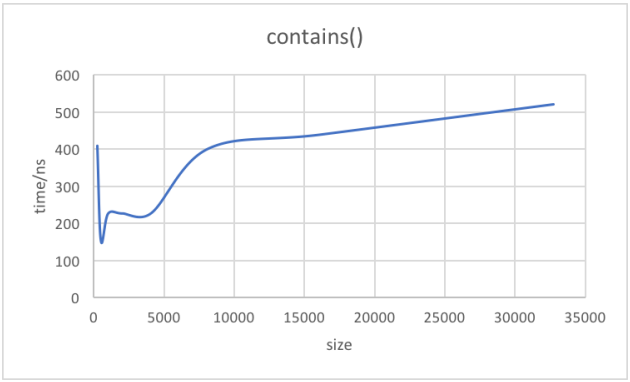
(1) The initialization is different since the sorted set should take comparator as a parameter to construct the class. And my sorted set class cannot get a specific element by like arr[i].
(2) No, because every time it needs to create a new class and the user need to pay attention to type casting sometimes.

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

(1) O(logN) because binary search is used to search for the element.

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?

(1) The graph is shown below, which matches the Big-oh behavior as it is like a log graph when size increases.



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)?

(1) The plot is shown below. Worst case: O(N+logN)≈O(N)