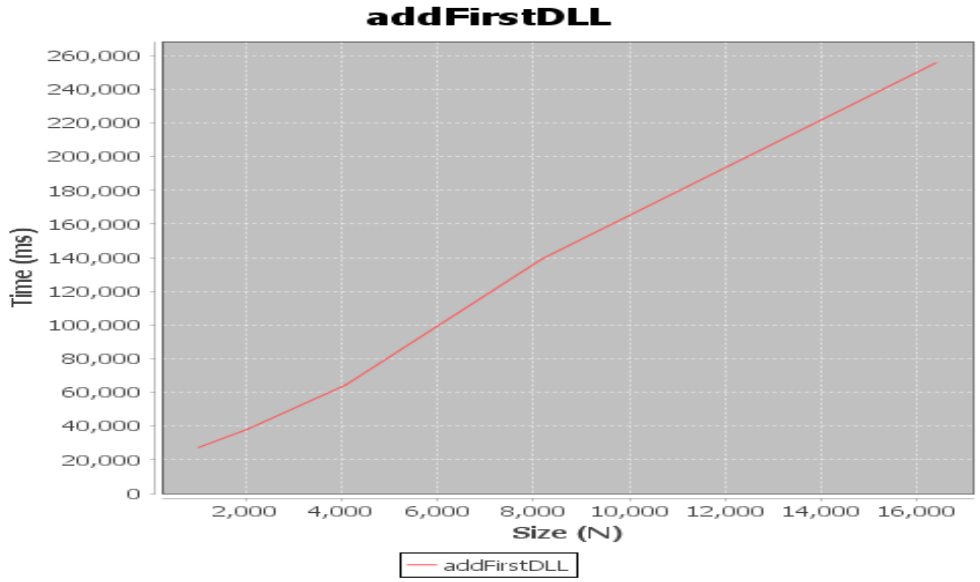
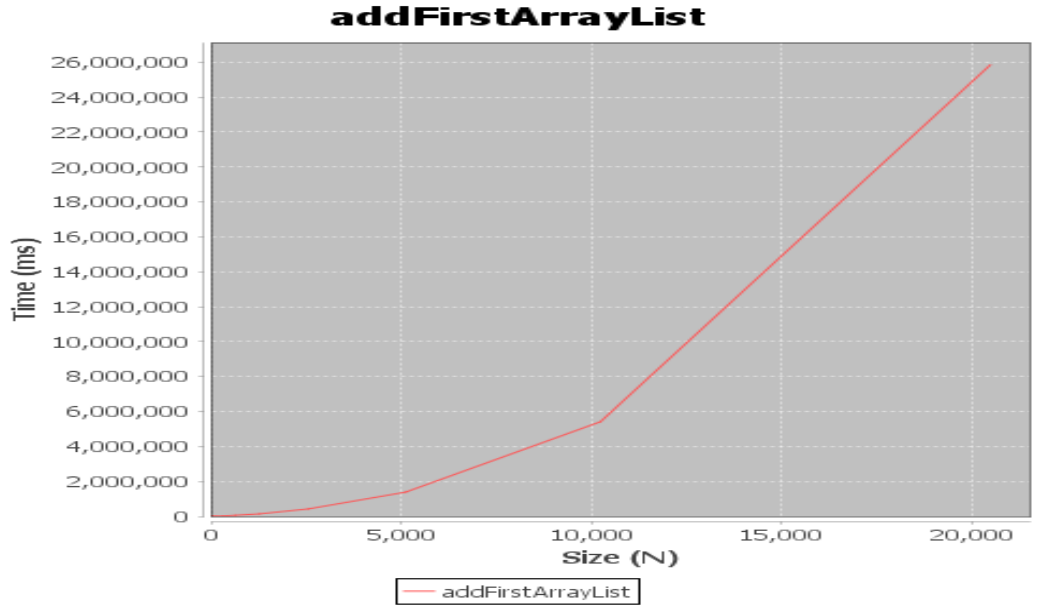
Torin McDonald
Analysis06

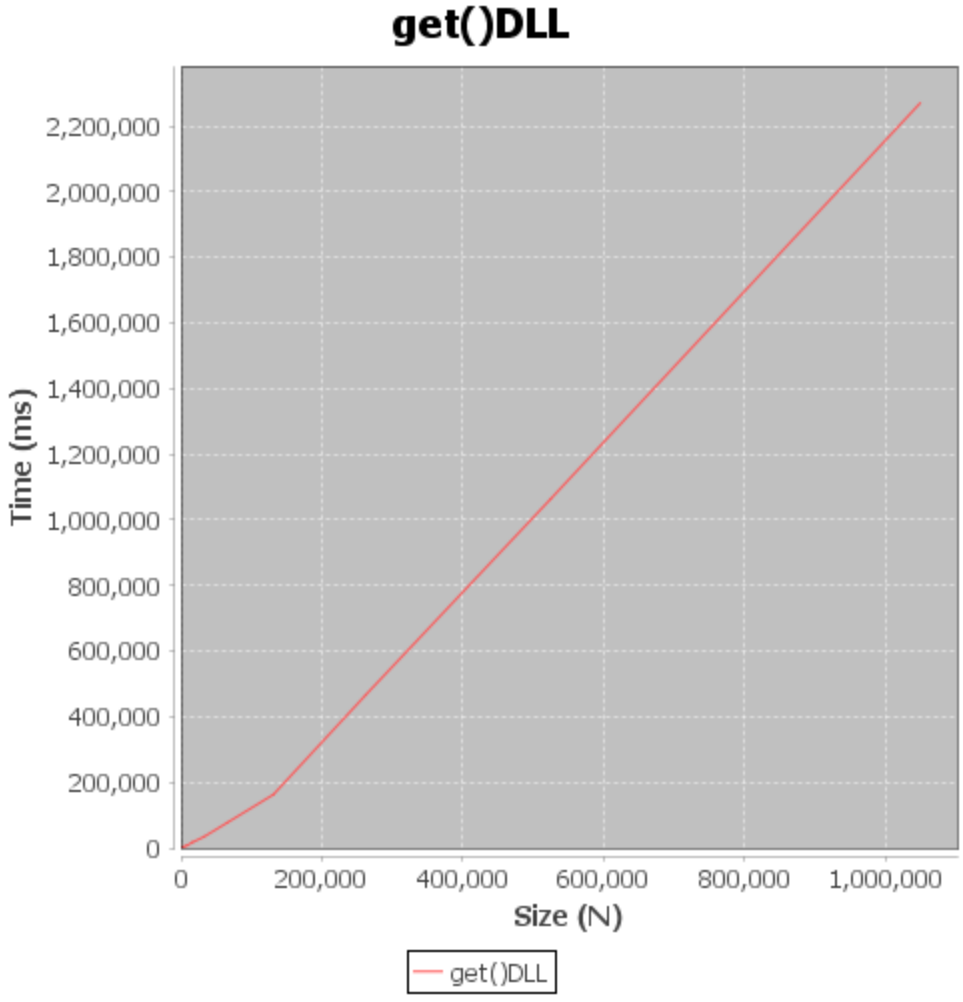
1) The behavior of addFirst was O(N). The graph below exhibits the total time to add all the elements of a particularly sized list. Its linearity indicates that the actual function addFirst is constant no matter what the size.



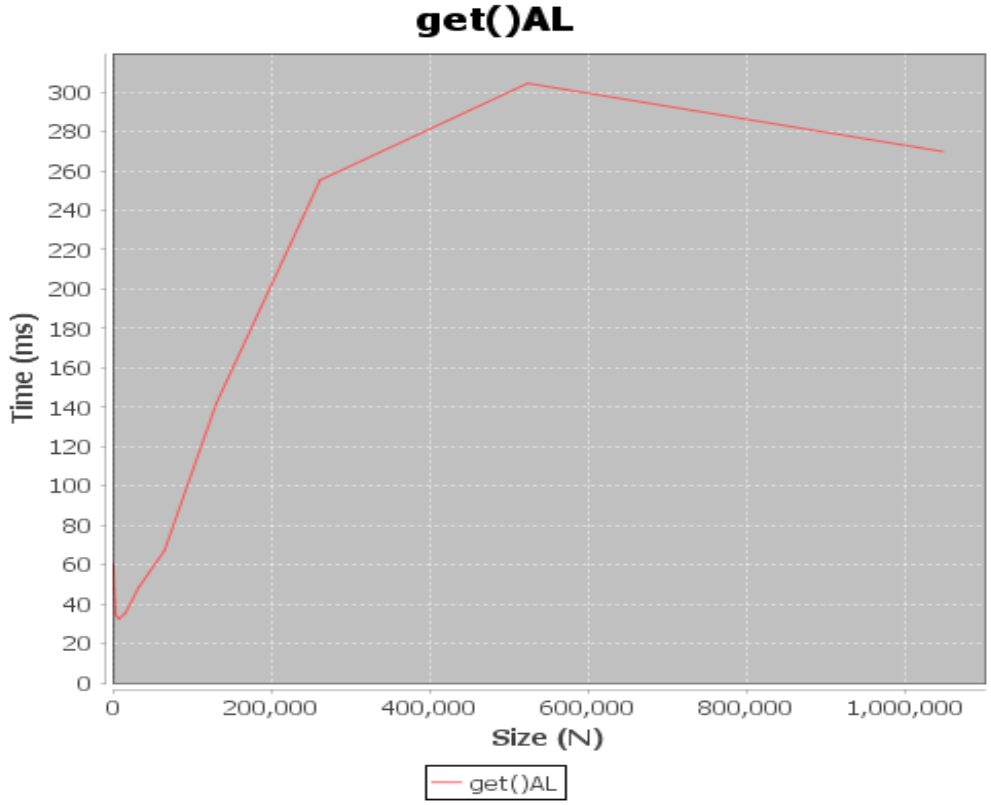
When this is compared to ArrayLists add(0,i) method, it is clearly a less complex algorithm. The graph of add First for Array list increases exponentially (again it is showing the total time to add all the elements into an ArrayList) indicating that the method has a complexity of O(N).



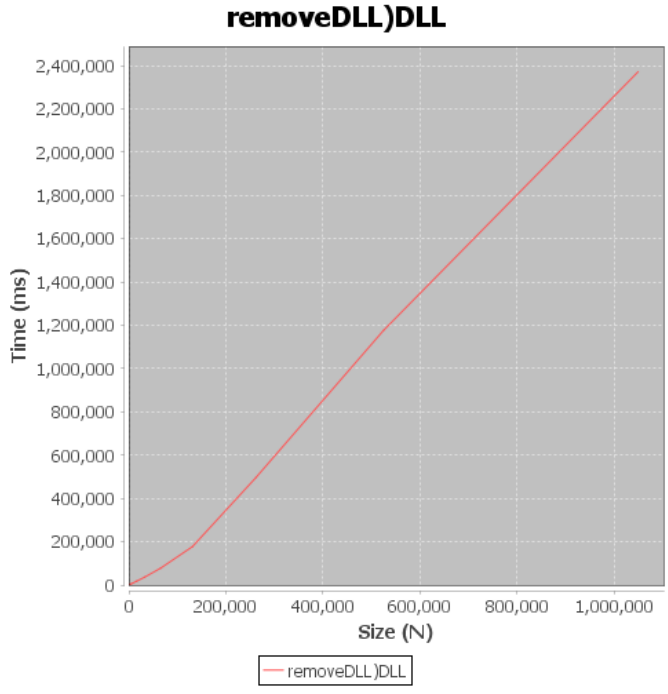
For DoublyLinkedList's method get(element), the complexity turned out to be the expected O(N).



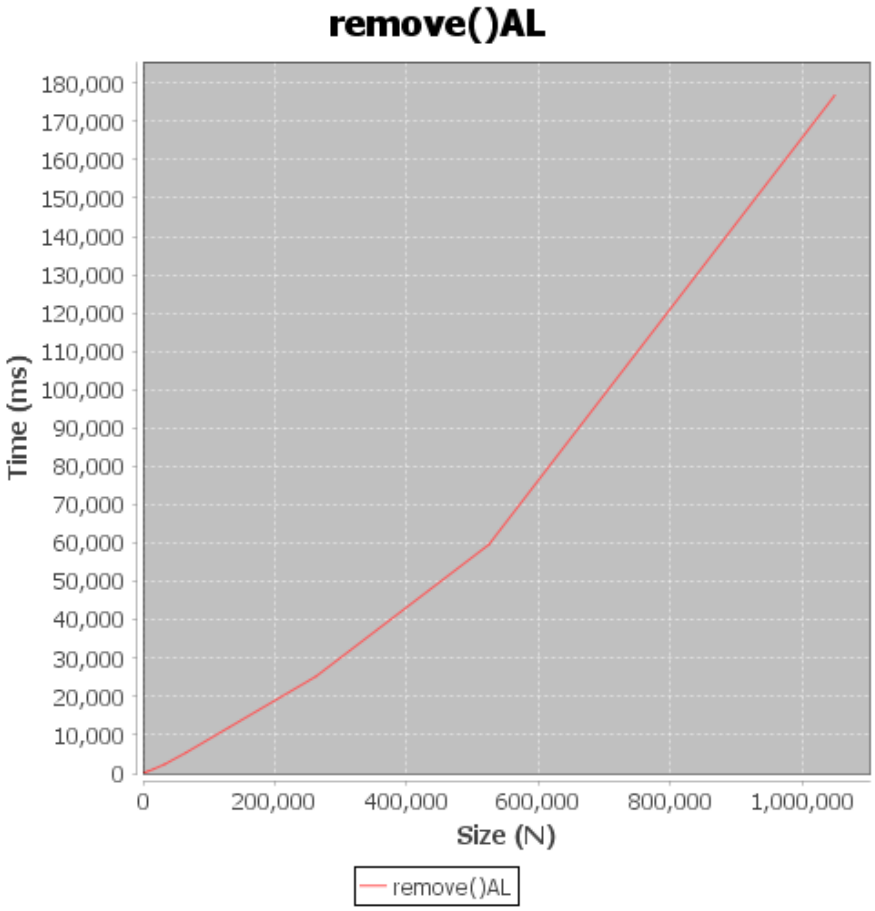
This is a worse complexity than Java's ArrayList method get(), which is O(N) (for large sizes).



For DoublyLinkedLists's remove method, the expected complexity O(N) turned out to be the reality.



The complexity for ArrayList's remove method is also O(N).



2) There are advantages and disadvantages to using ArrayList versus DoublyLinkedList. As exhibited above, there are some methods (addFirst, addLast, ect) whose complexity is much better in the DoublyLinkedList than in an ArrayList. Alternatively, methods like ArrayList's get have better complexity than those in DoublyLinkedList because they don't have to traverse the array. Finally, there are some methods like remove which have the same complexity in either case.

3)Java's LinkedList performs very similarly to my DoublyLinkedList. The main difference is that LinkedList traverses the list either at the start or from the end depending on which is closer to the element you're searching for.

4)Using an ArrayList versus a LinkedList would change the functionality of BinarySearchSet depending on the method. It would change the add/remove/contains methods in the same way exhibited in question 1.

5) I spent 15 hours on this assignment.