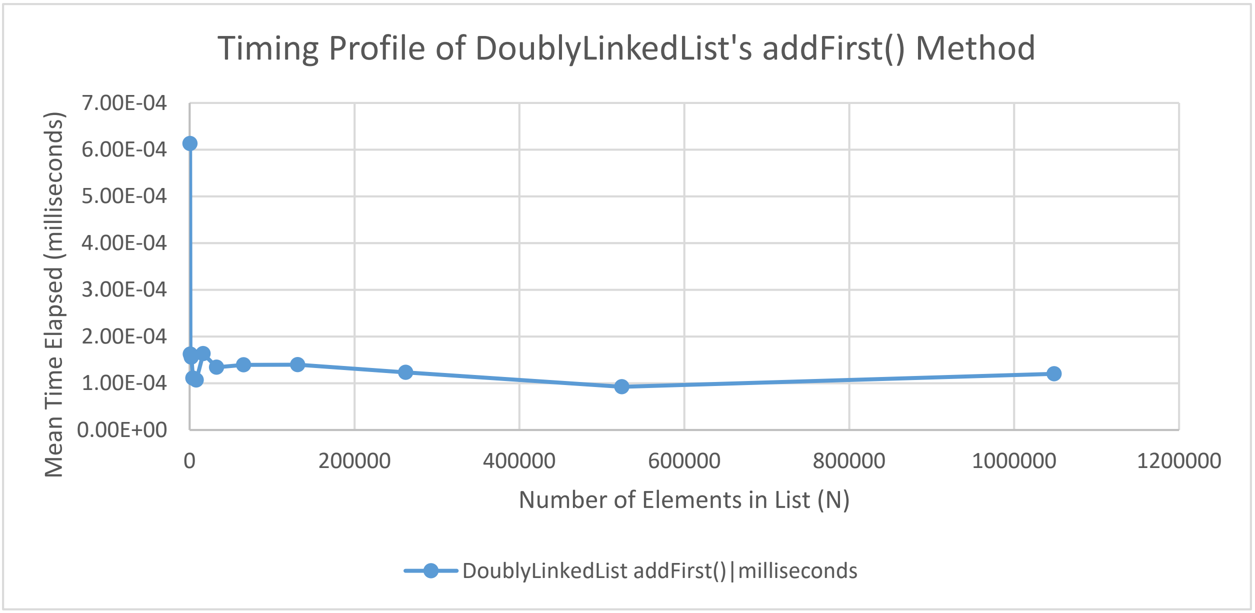
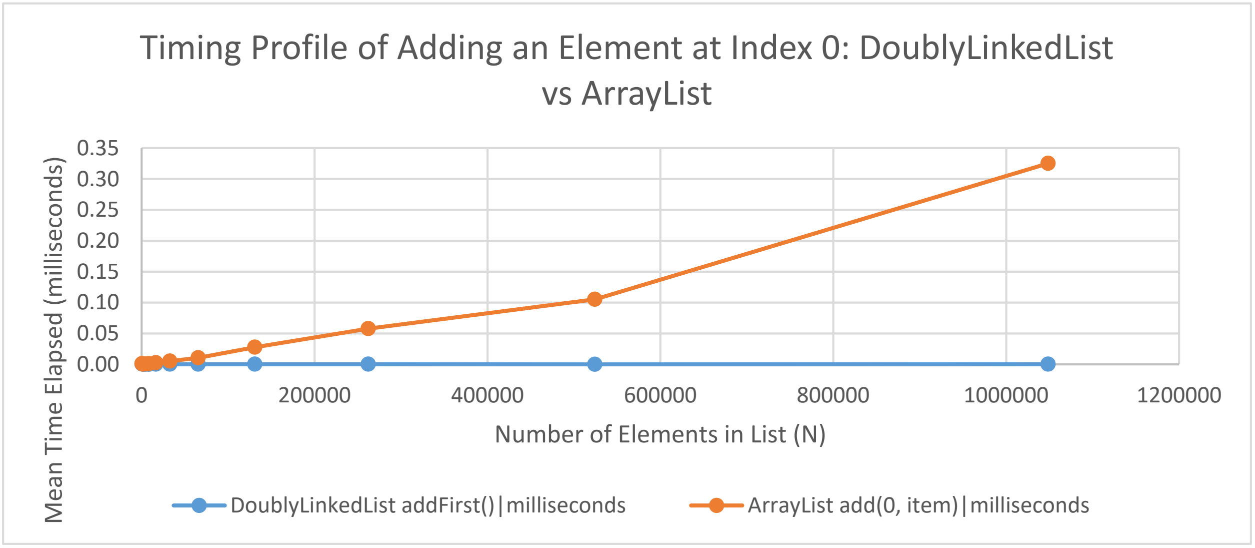
CS 2420
Tyler Adams
Assignment 6
Analysis Document

1. Yes, the running time of the addFirst() method is evidently O(c) as I expected. All of the following plots were averaged over 1000 iterations. Please view the plot of running times for addFirst():

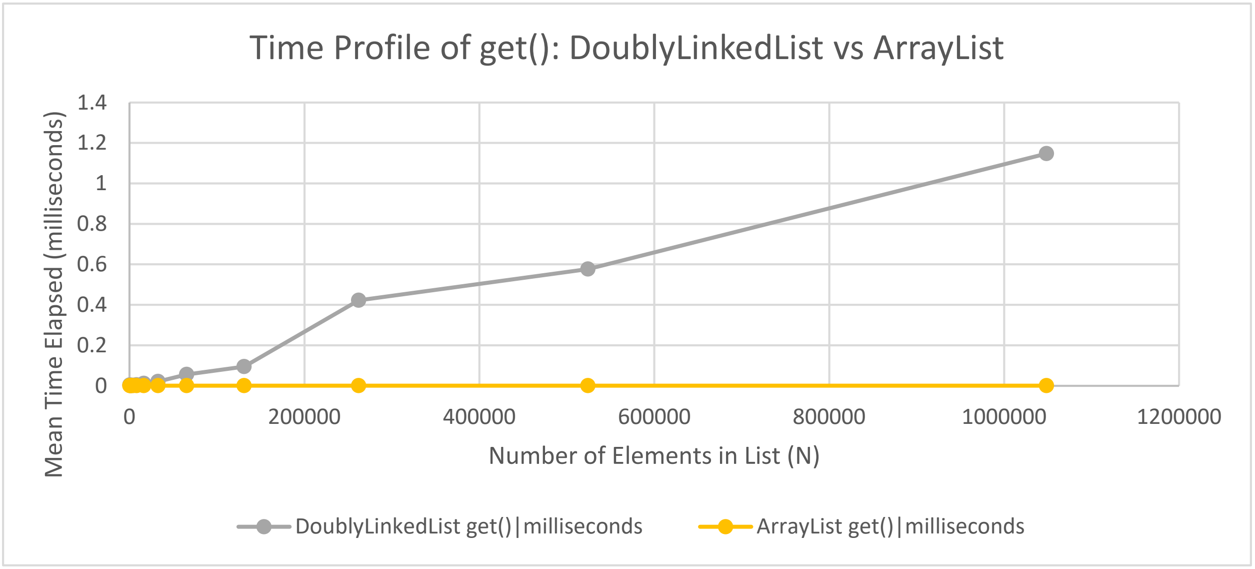


In comparison to the add(0, item) method of the ArrayList class, addFirst() is much faster:



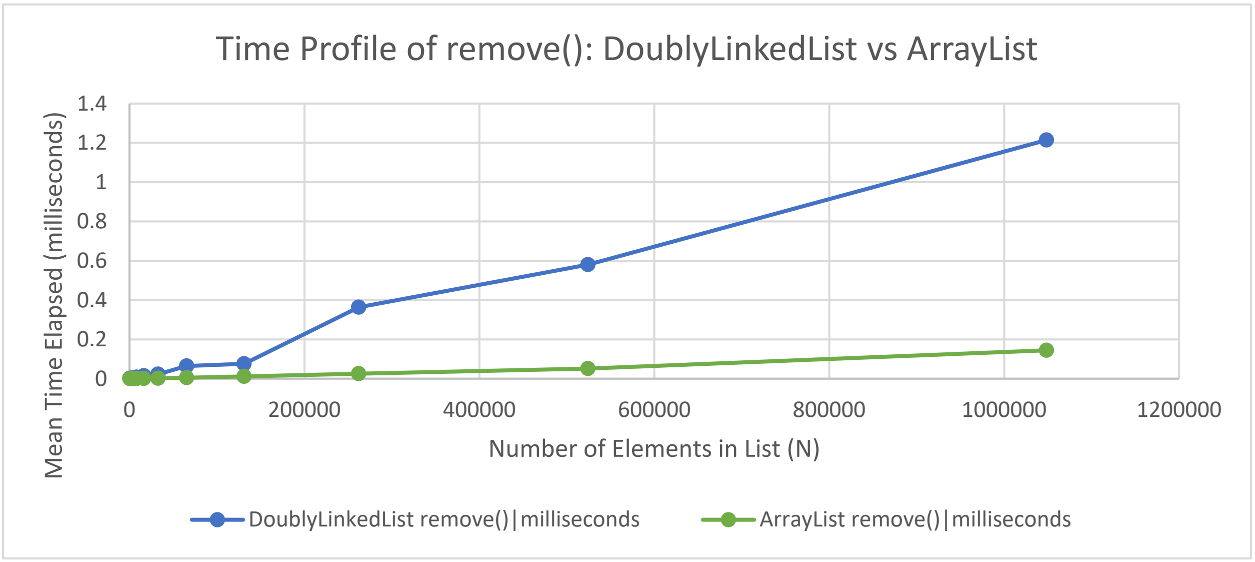
The complexity of ArrayList's add(0, item) method is clearly at least O(N) as expected, and the time elapsed quickly exceeds that of addFirst() in the DoublyLinkedList.

The timing of get() was not surprising either. For both the get() and remove() time plots, I choose an index with a pseudo-random number generator every time (without timing it of course), and again averaged the time across 1000 iterations. Consider the chart of get():



DoublyLinkedList's profile appears linear and ArrayList's profile appears constant as expected.

As for remove(), I was surprised that ArrayList completed the method more quickly, but at least they both appear to be linear:

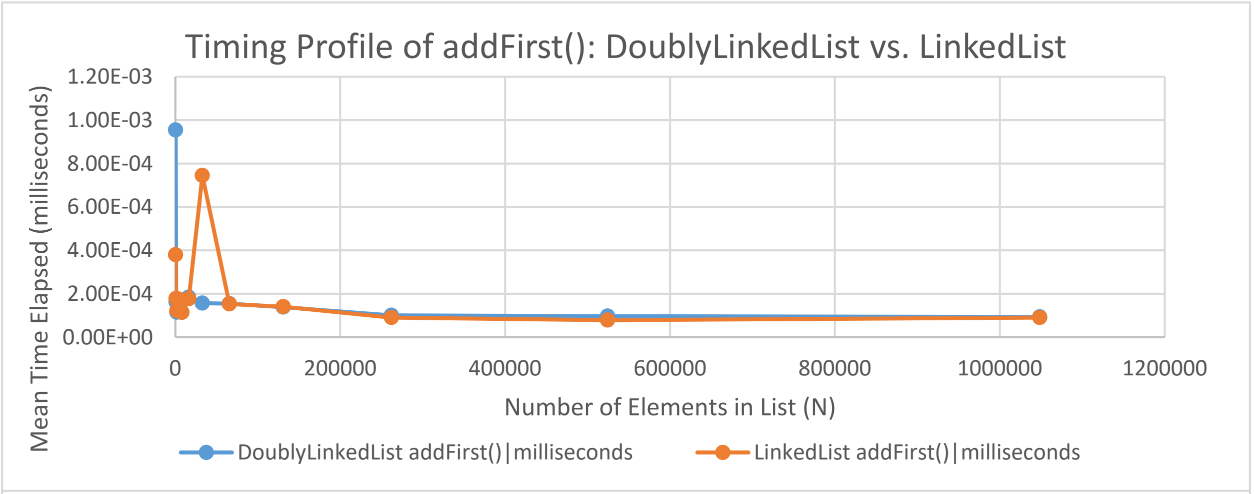


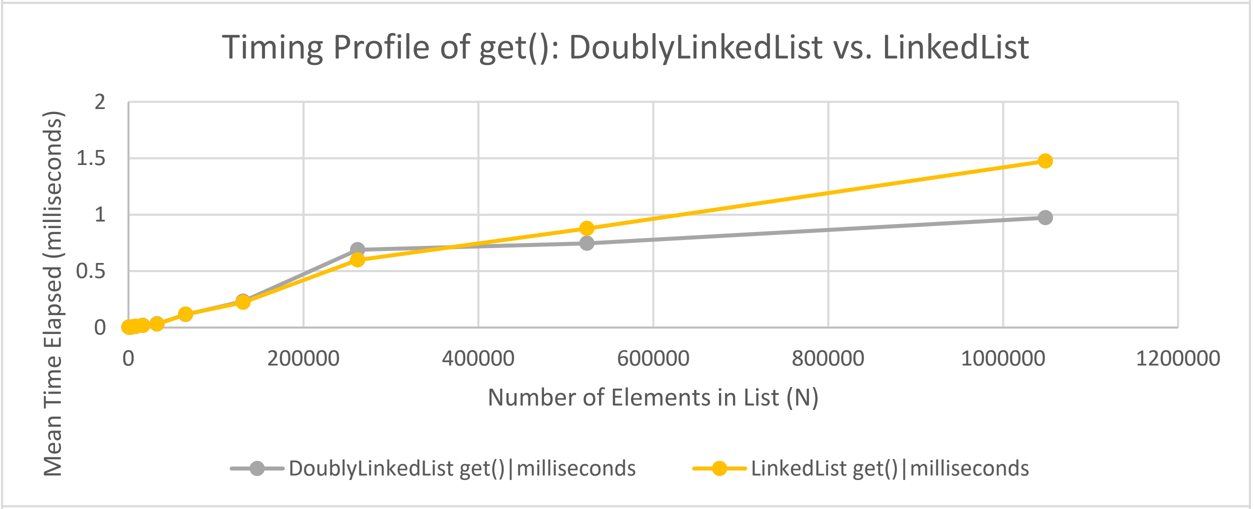
In summary, the apparent complexity of DoublyLinkedList's addFirst() method is O(c), ArrayList's add(0, item) method is O(N), DoublyLinkedList's get() method is O(N), ArrayList's get() method is O(c), DoublyLinkedList's remove() method is O(N), and ArrayList's remove() method is O(N).

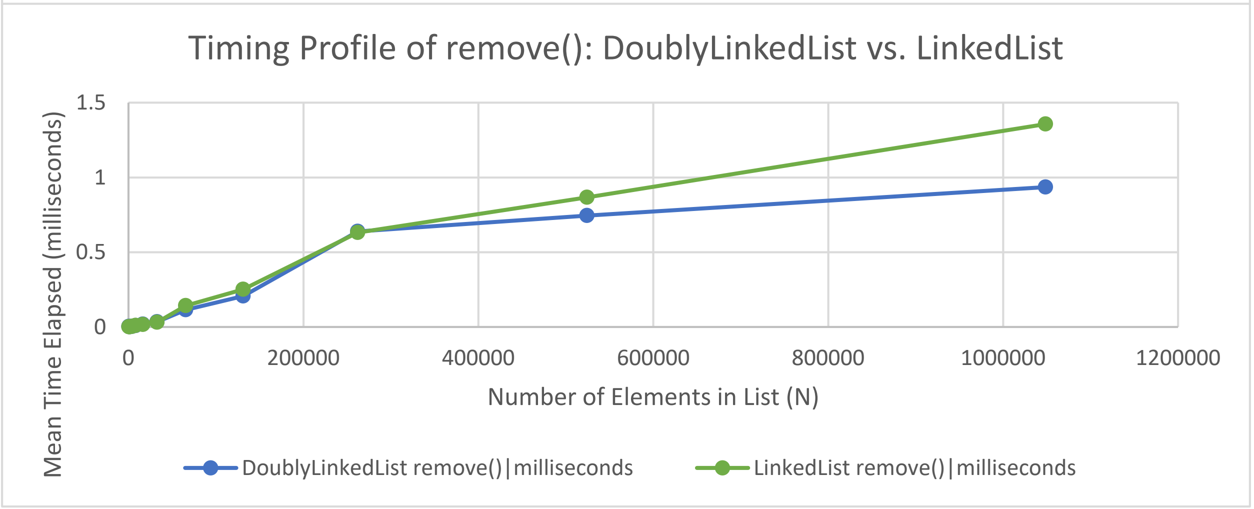
2. DoublyLinkedList's functionality is very similar to that of ArrayList, but the lists are not quite interchangeable. The main difference is that DoublyLinkedList uses slightly modified method signatures for things like addFirst() and addLast(), which ArrayList lacks. Also, DoublyLinkedList does not have a set() function. However, for the most part, DoublyLinkedList is functionally

equivalent to ArrayList. If the user doesn't add many elements toward the beginning of the list, ArrayList generally performs more quickly than DoublyLinkedList in most of its procedures.

3. As far as I can tell, the functionality of DoublyLinkedList and Java's LinkedList is identical, apart from LinkedList's implementation of Collection and several other extra classes. The performance is also very similar as you can see by the running times of the addFirst(), get(), and remove() methods of each:







However, unfortunately for Java, it appears that DoublyLinkedList performs slightly faster than LinkedList on the get() and remove() functions. Why yes, that is what I said.

4. A LinkedList would significantly impede the performance of the procedure of a binary search, because the binary search must repeatedly locate the midpoint of a subarray (or sub-list in this case). So an ArrayList would allow the search to be much quicker for finding an element. Now, a LinkedList might provide advantages in removing and adding elements on small sets, but with the compromised performance of the binary search, it might not provide any advantages at all.
5. I worked for about 7 or 8 hours on this assignment.