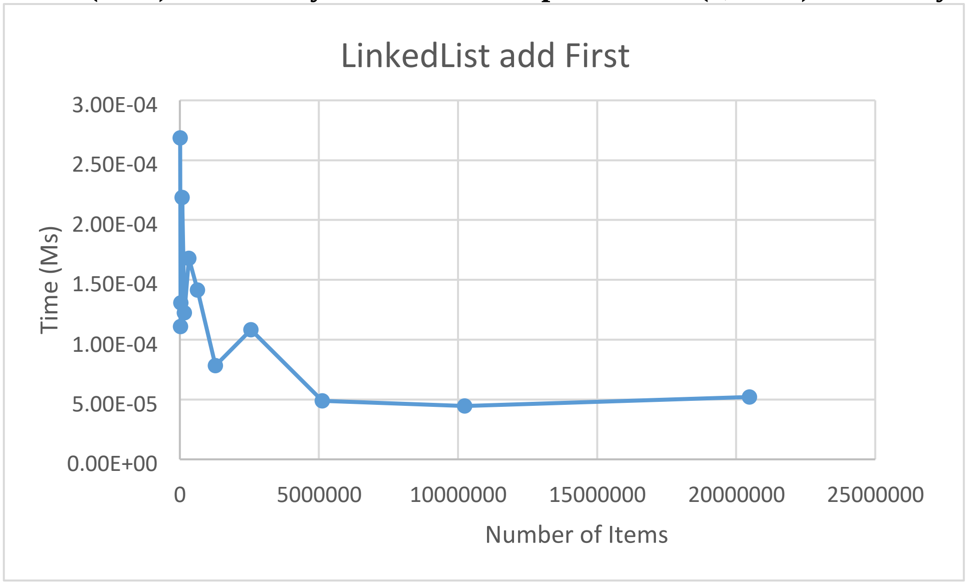
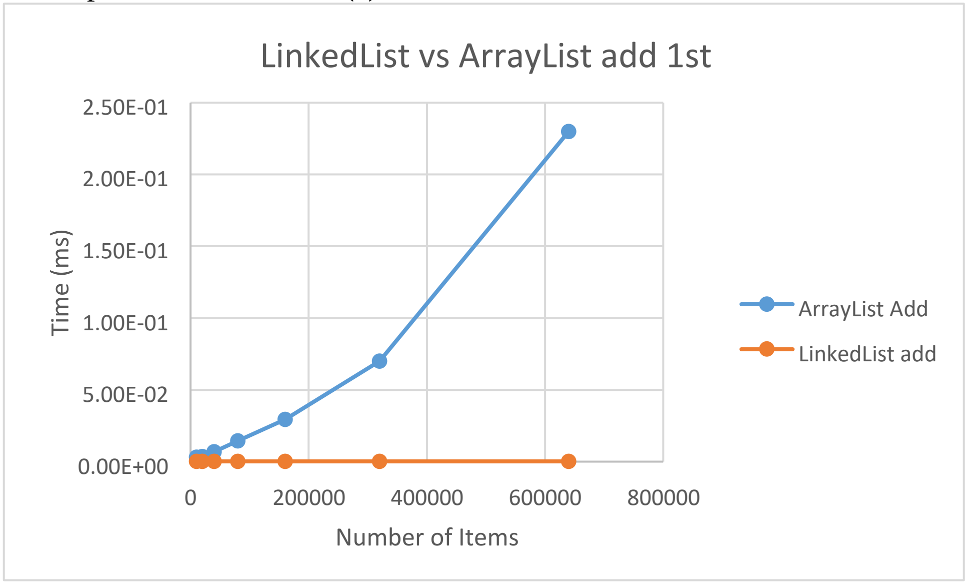
Dylan Northcutt
U1055102

1. Is the running time of the addFirst method O(c) as expected? How does the running time of addFirst(item) for DoublyLinkedList compare to add(0, item) for ArrayList?

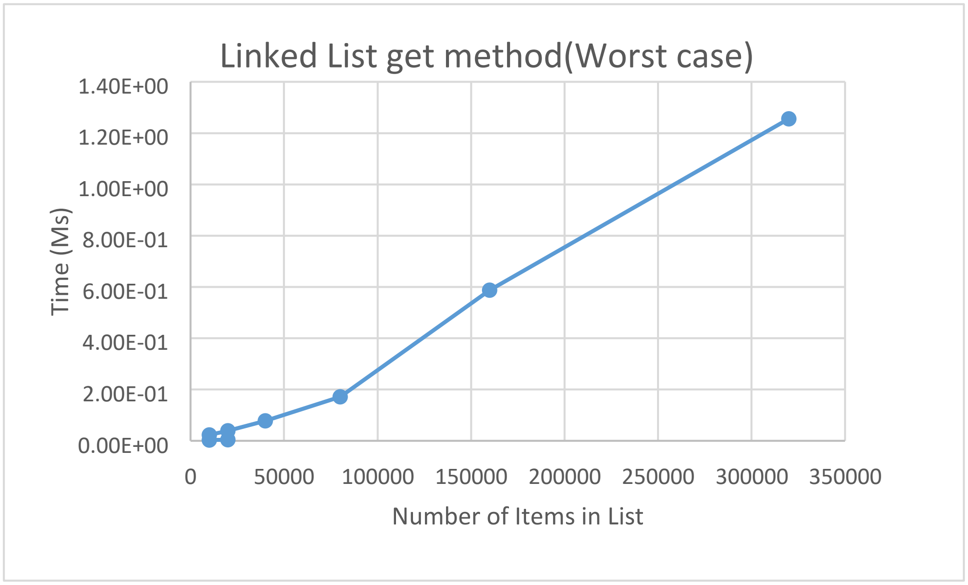


Aside from the very start where the runtime is random, the runtime does become constant. This does show the expected runtime of O(c).

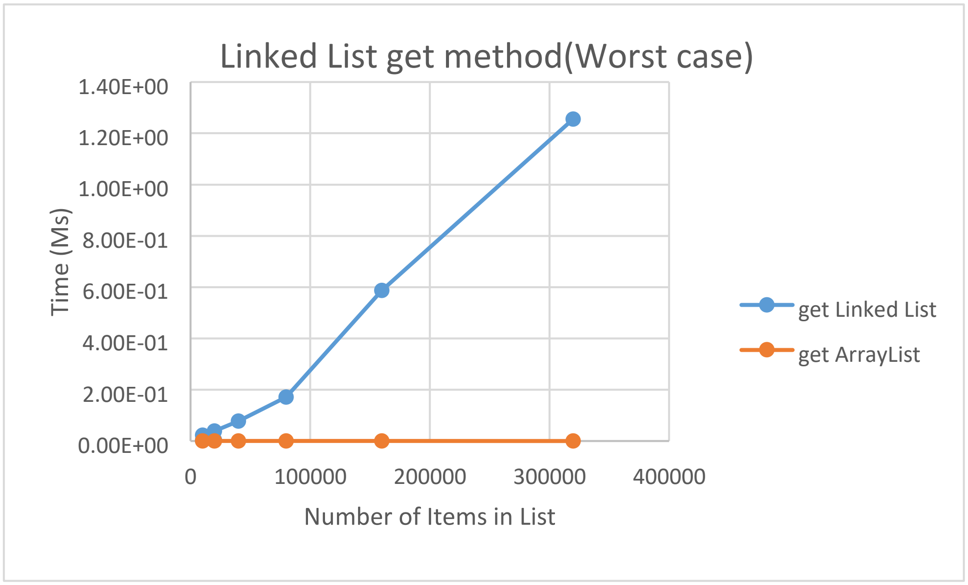


The runtime of the addFirst method is incredibly small compared to adding first in an array list. This is because the runtime of addFirst is constant as it takes the same time to assign different next and data values to head while the arraylist has to move all the items over for an item to go in the first space.

Is the running time of the get method O(N) as expected? How does the running time of get(i) for DoublyLinkedList compare to get(i) for ArrayList?

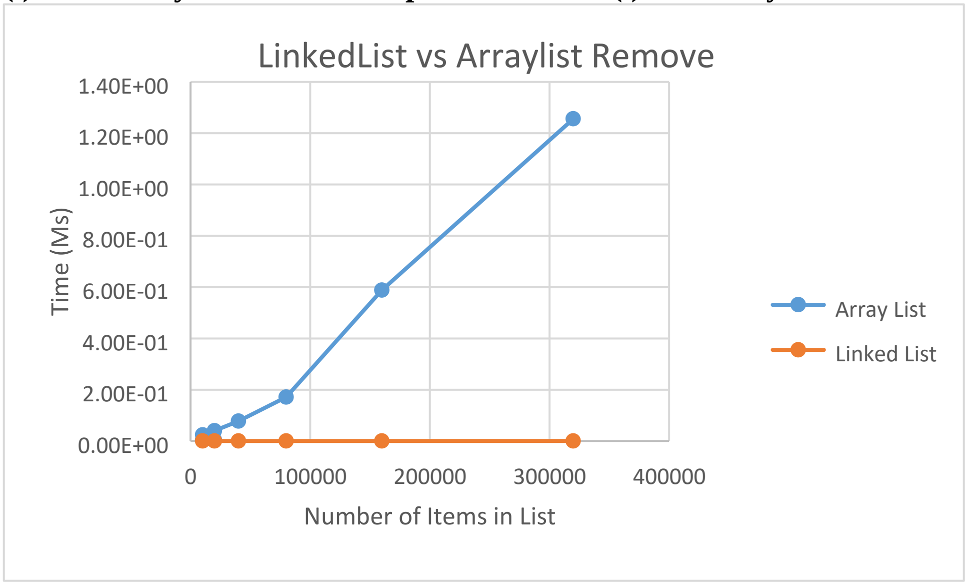


The runtime of get for the linked list does match the expected runtime of O(n). the charts show the runtime for the worst case of the get method.



This shows the opposite data as the add first method. This is because to get an item in an array only one simple addition is needed but to get an element in the LinkedList each item needs to be checked until the item is found. The runtime of get arraylist is O(c)

Is the running time of the remove method O(N) as expected? How does the running time of remove(i) for DoublyLinkedList compare to remove(i) for ArrayList?



The performance of the remove method is the same as the add method because basically the same thing is happening except instead of putting an element in the new slot we are taking one. The LinkedList is constant because all that is being done is having the next/prev reassigned while in the arrayList all elements are being shifted.

2. **In general, how does DoublyLinkedList compare to ArrayList, both in functionality and performance? Please refer to Java's ArrayList documentation.**

In general, both ArrayList and LinkedList are very similar. They both allow to add, remove and check items. They differ in their strengths and weaknesses in doing these. LinkedList is stronger in adding or removing elements while ArrayList is stronger in retrieving elements. They both function similarly but they perform better in opposite situations.

3. **In general, how does DoublyLinkedList compare to Java's LinkedList, both in functionality and performance? Please refer to Java's LinkedList documentation.** Our implementation of DoublyLinkedList is similar to javas linked list in several methods. The DoublyLinkedList is different in that the list can be traveled in both directions, meaning it has both next and prev. LinkedList has several methods that we created in DoublyLinkedList For example, there are two add methods for adding to any place in the array and adding to the end. There is also an addFirst Method as well as clear and get. Both the DoublyLinkedList and java's LinkedList are very similar.

4. **Compre and contrast using a LinkedList vs an ArrayList as the backing data structure for the BinarySearchSet (Assignment 3). Would the Big-O complexity change on add / remove / contains?**

An arraylist is more practical to use in a binary search. This is because in searching it is more common to be getting elements rather than adding them. Getting elements is where arrayList is better than LinkedList. LinkedList are more useful if adding items is more common than retrieving them.

5. **How many hours did you spend on this assignment?**

I spent about 7 hours on this assignment.