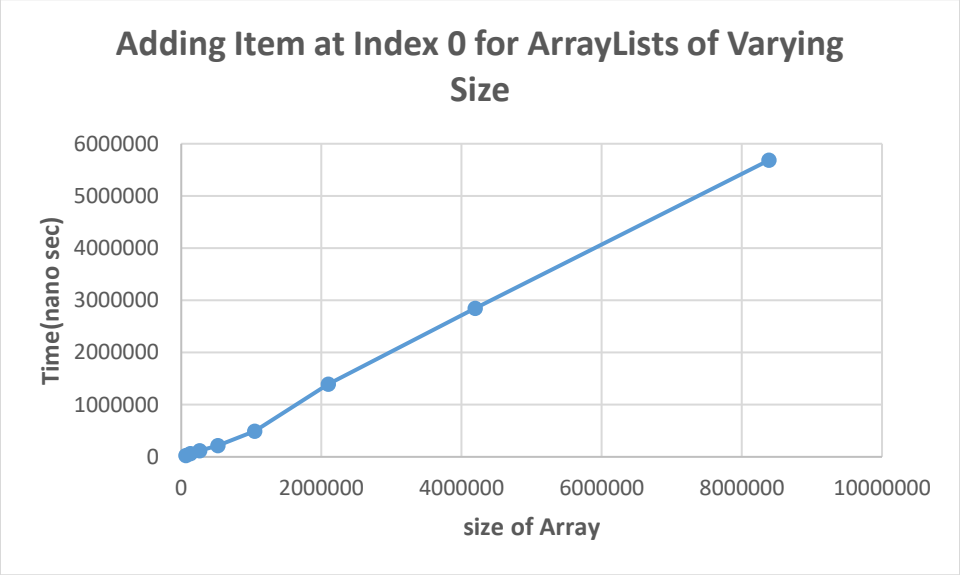
Brian Park

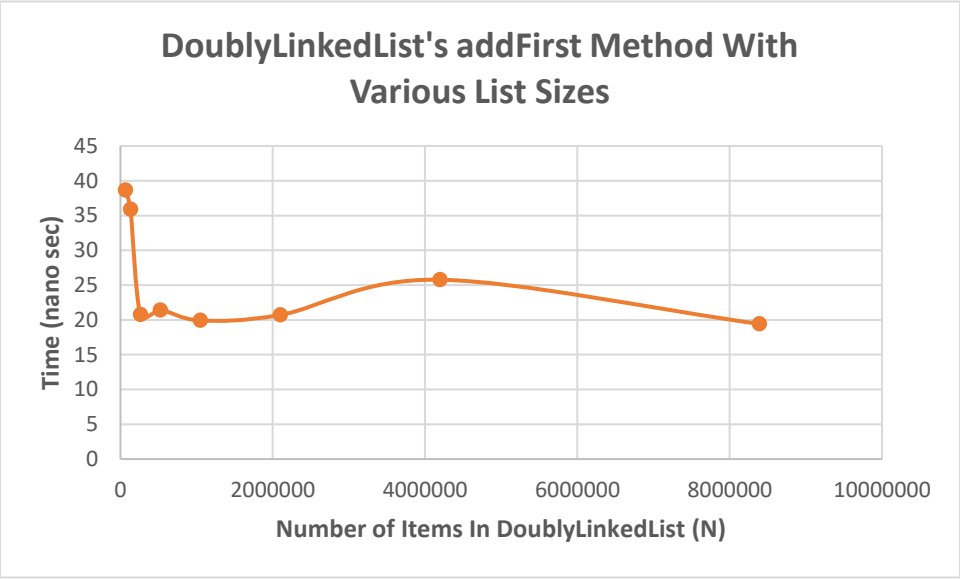
Cs2420

Assignment 6

1. The time of the addFirst method is constant time as expected. This is expected as the only instructions that are being executed are changing the references of the first item in the list. Conversely when adding an item to the beginning of an ArrayList, because of the nature of the array, all of the items must be shifted forward one index. Because of this the big O behavior as the number of items in the array becomes N. As expected the time it takes to get an item from the linked list is of order N. This method is where ArrayList out competes a linked list as when passing an index, the array knows exactly where to get the item from, resulting in the get method for ArrayList being constant time, much faster than the linked list's order N speed. The remove method is more interesting as the order of both ArrayLists and linked lists are N. This is because the linked list has to iterate through the list until the specified index. The ArrayList jumps to the index, then removes it by moving everything else forward. Both resulting in a bigO of (N).



Note the linear growth and time of each point.

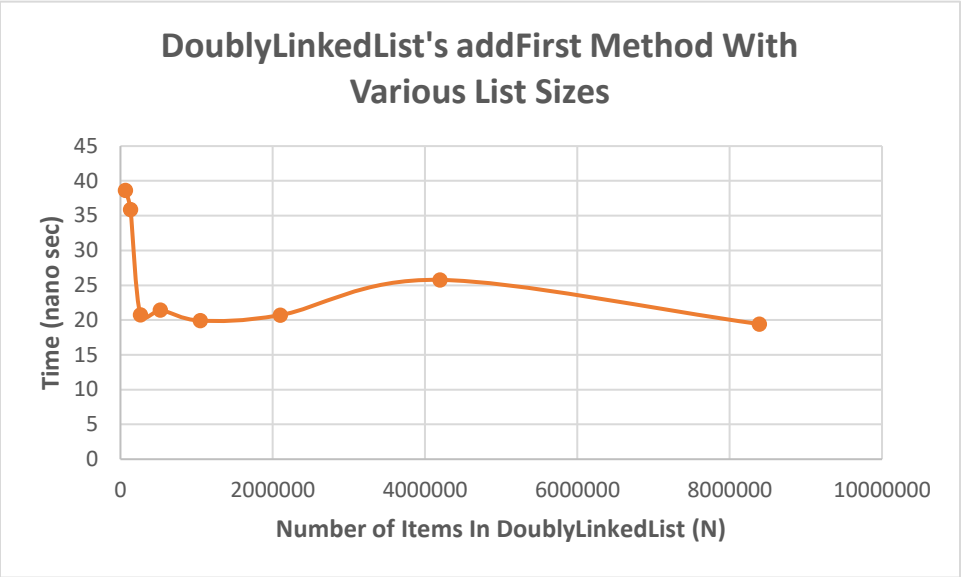


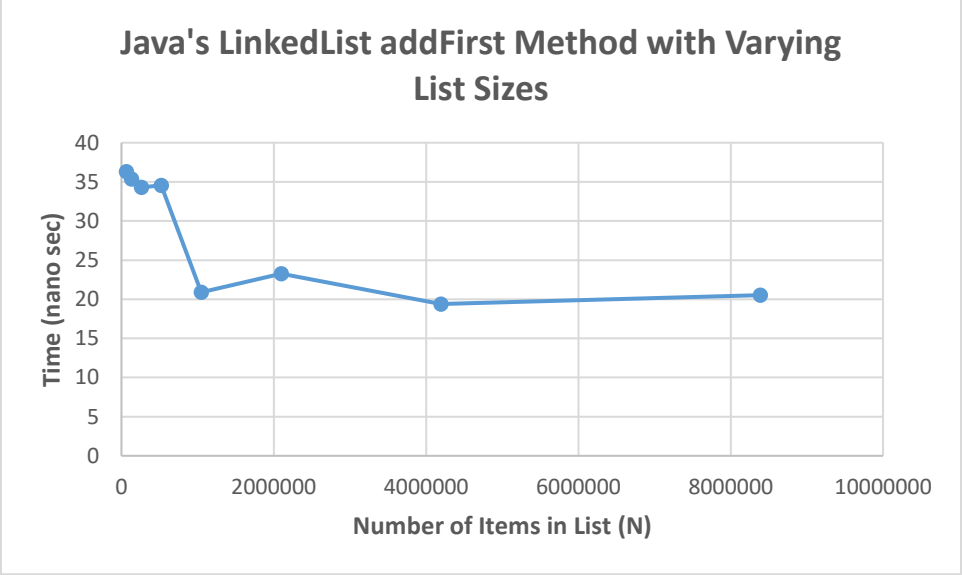
Note the time, DoublyLinkedList method call remains a flat trend.

2. For the most part, when managing anything but specific lists, I would favor an ArrayList over a linked list such as ours. This would depends on what the list was being used for. If the list was being iterated over and things were being removed and added at specific points, the clear winner would be a linked list. However, for just storing and accessing data, the ability to get and set places in the list that the ArrayList possesses is very useful. In the case the list was being iterated over regardless and elements were being analyzed, added, or removed regardless, a linked list would be much more useful in terms of performance in this case. This is also true if we were expanding a list in both directions as adding elements to the front of a linked list is O(c) as well. Over all in my opinion ArrayLists are on average more useful in terms of performance, unless performing the above advantages of linked lists frequently.

3. In terms of functionality, DoublyLinkedList and Java's linked list are fairly similar. Java's version is capable of behaving as a stack and has a pop method, peek methods and several other helpful methods that our iteration lacks. Aside from all of the extra utiliy that LinkedList brings, at their cores they are very similar. In terms of performance, java's linked list's methods have been tuned in many ways according to information available. In terms of bigO behavior they are

similar. It is very difficult to compare their constant speeds as behind the scenes problems like garbage collection ect skew the data, however they are very similar.





Notice similar performance, often within nanoseconds of each other.

4. It is interesting to compare and contrast if BinarySearchSet would be better off backed with a an arraylist or array. Over all it is my belief that using a ArrayList would be a better option for performance but this would depend on what the set was for. Because BinarySearchSet relies on iterators, in some cases, the Linked list would shine in these areas. When an item is being added for an array backed BSS, we use Binary search to find its location, and then we insert it into that

location by moving everything else forward. Which is an N+logN behavior while if we iterated through the array, found the insertion point and added the item with a linked list, the action would be N +c. This is also the case for the remove method. As N gets large however this does not matter as the BigO behavior is N. The contains method is able to utilize binary search which has logN behavior while a linked list still must rely on its N behavior for contains. The getters and setter of the array backed BinarySearchSet would be much quicker than one that was back with a linked list. Over all each has benefits, but the “best” way to back a BinarySearchSet would depend on what you were doing with it.

5. I spent approximately 6 hours working on this assignment as it was much easier to be efficient when on a solo assignment.