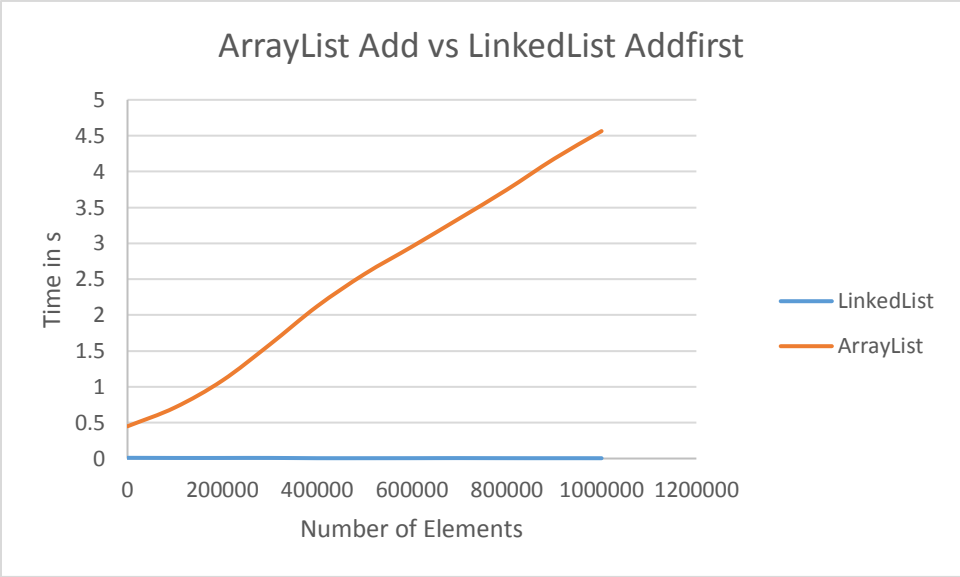
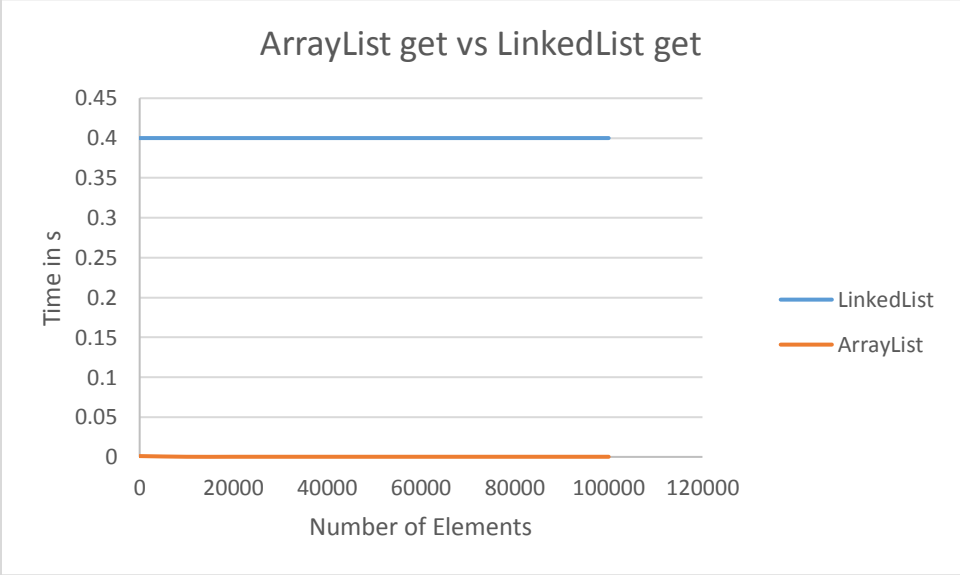
Jordan Gardner

U0566259

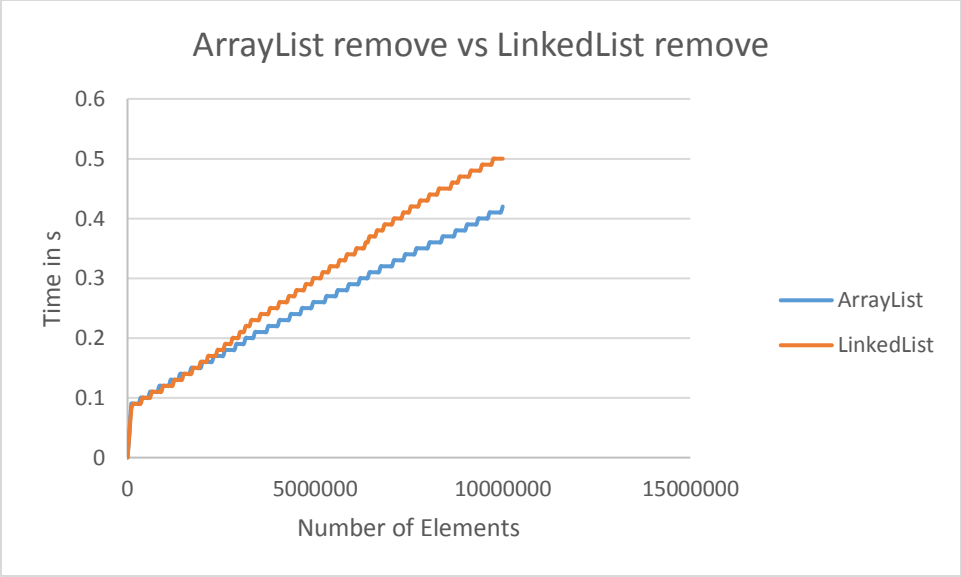
Analysis Assignment 6

1. 

This experiment does indeed show that the Big O notation for Linked list addFirst method is constant. It further shows that ArrayList add method is order N. These are the correct notations as presented to us in the lecture slides. I would also like to point out how much better LinkedList was in run time vs ArrayList.



Here is the experiment for the access methods. I did suspect that an ArrayList would do much better on this experiment than LinkedList. This is because you have to access item by item in a linked list in order to find what you are looking for. This is not the case in an ArrayList, you can access data much more efficiently. This graph shows the get method for Arraylist which is very very fast compared to the LinkedList method. This graph even under scrutiny shows that Arraylist runs in constant time for its get method while Linkedlist is much slower at N.



This is the graph of the remove experiment. I have shown here that the order for LinkedList is indeed N. Furthermore, I show that for Arraylist it seems to be N as well. I think this is an accurate analysis seeing that in order to remove from a linked list you have to switch the variables that access it; and in order to remove an item from an arraylist you have to shift all other items down. This would give the effect of each having an order N notation.

2. According to the Java API on Arraylist most of the main methods run in constant time and are favorable to linkedlists. It does say however that the add method runs in amortized constant time which means as you add n elements it takes n time. I would have to agree that Arraylist has outperformed LinkedList especially on the get method and other accessibility methods. However, on the add method LinkedList does outperform and might serve useful in situations like queues or stacks where you only want to add or remove from the top of the list.
3. It seems as if Javas got us covered on this one as well. Their LinkedList class behaves as our DoublyLinkedList class does. It has all the methods we used plus more. It specifically says in the LinkedList description that *"All of the operations perform as could be expected for a doubly-linked list. Operations that index into the list will traverse the list from the beginning or the end, whichever is closer to the specified index."* https://docs.oracle.com/javase/7/docs/api/java/util/LinkedList.html
   As far as performance and run time. I would suspect their program to be about the same maybe slightly better than this homework assignment was designed to be.
4. Assignment 3 uses BinarySearch and an ArrayList to back the data. This has a general performance of logN due to binarysearch. In worst case this becomes N which is still in a tie with LinkedList for get and remove. LinkedList does have great performance with add and in contrast to assignment 3s which could turn out to be order N. With all this being said an Arraylist combined with binarySearch may prove to be more worthwhile in dealing with lists you need access to. Using binarysearch allowed us to quickly access data that would be impossible to do in LinkedList in comparable time. Side by side comparisons Linkedlist wins on add, arraylist with binarysearch wins on get, contains, and remove.
5. 13hours