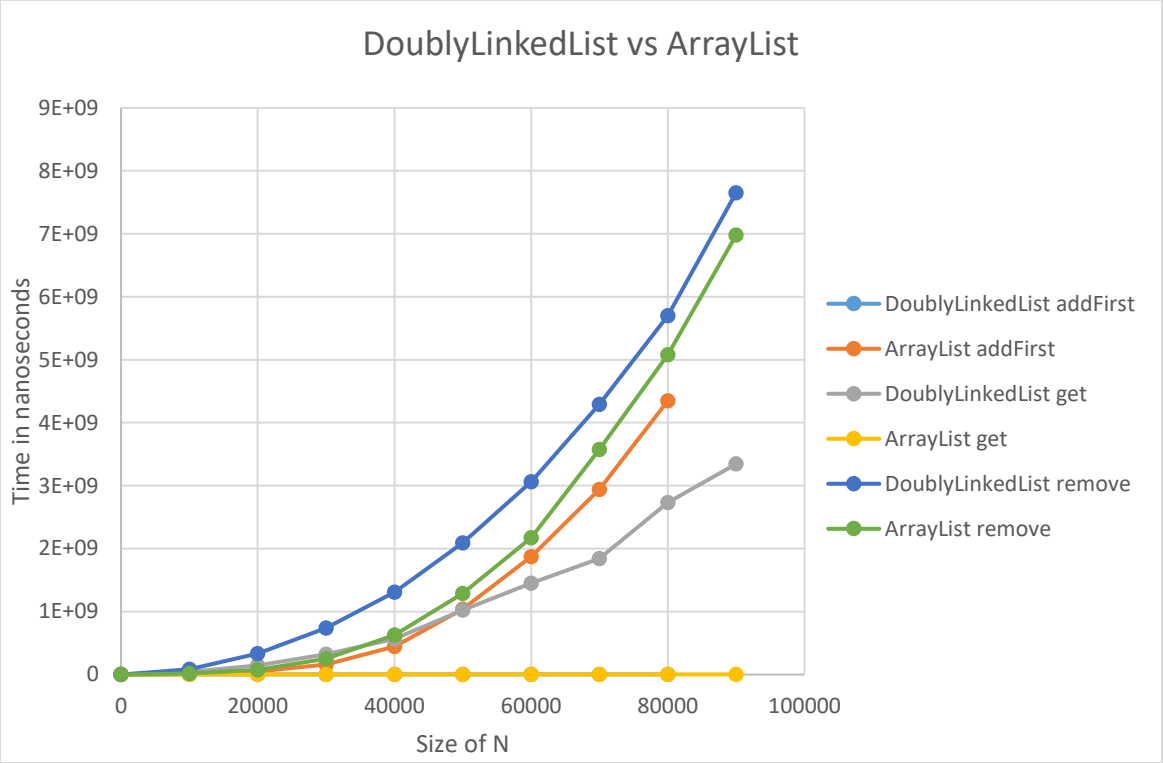
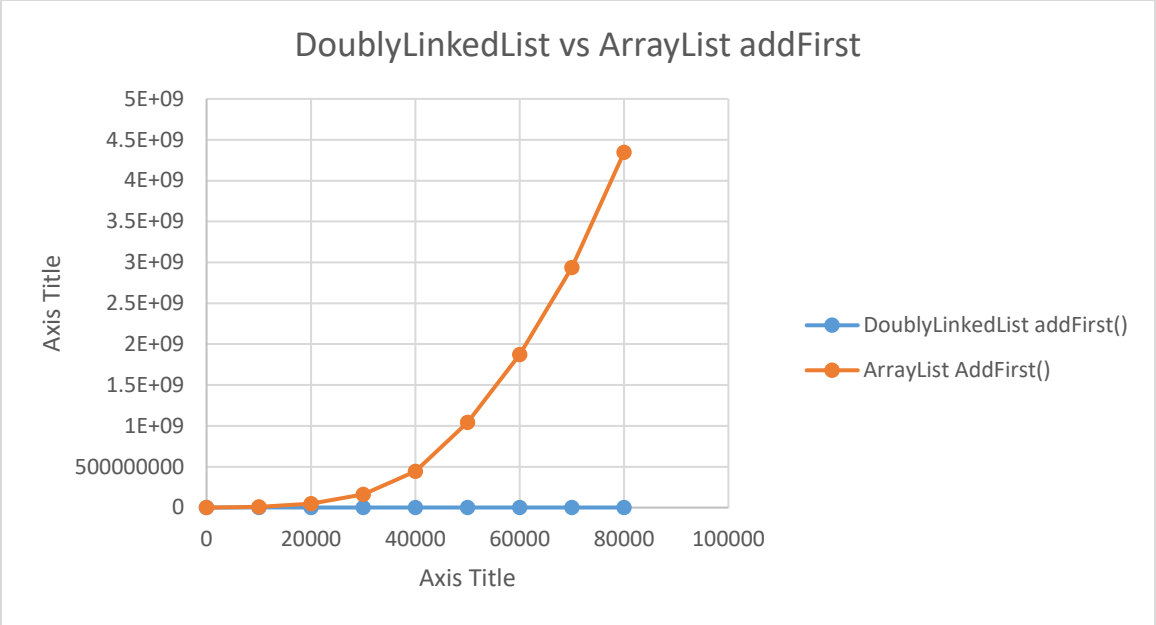
Chenxi Sun

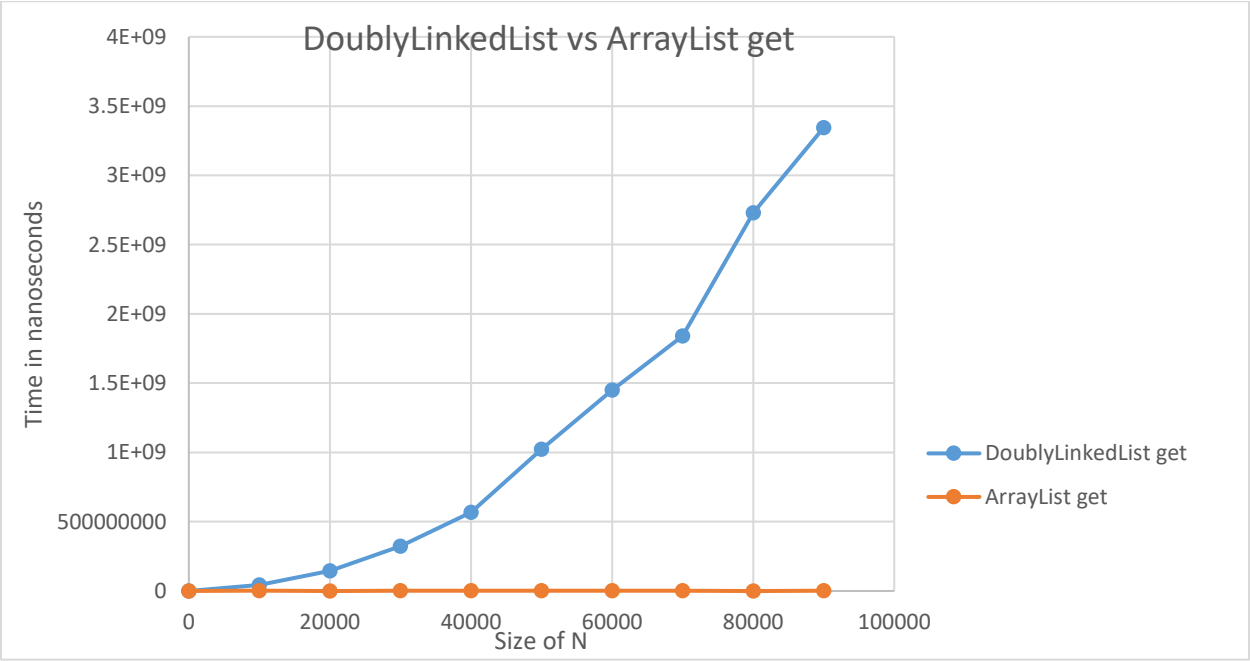
Assignment 6 Document Analysis



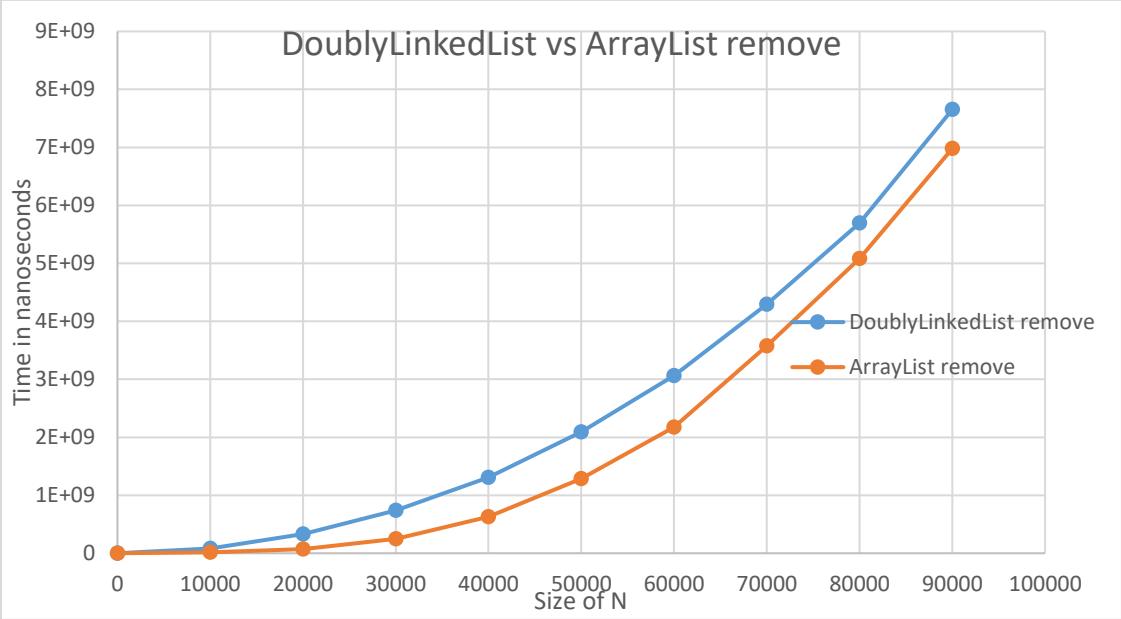
1. The big O of addFirst() for DoublyLinkedList is O(1). The reason is that to add the first item to the DoublyLinkedList all you have to do is reassign the header node to the added item. For ArrayList the add first method which is basically add(0, item), will have the big O of O(N). This is because that while get for the ArrayList is O(1), the added item at the first spot will shift the array backing the ArrayList forward 1 spot creating a O(N). So for addFirst(), it is much better to use a DoublyLinkedList rather than a ArrayList. This is show in the chart below as addFirst() for DoublyLinkedList shows a constant big O, while for Arraylist it shows a linear big O.



For the get() method it is much better to use an ArrayList. The big O for Arraylist for get is just O(1), as it times the index by a bit value to get the value at a specific index. For the DoublyLinkedList the get method is much more time consuming. The linkedlist has to iterate through the list index amount of times to get the value at that specific node. This creates a big O of O(N) for linkedlist. Since this is a DoublyLinkedList the get method is optimized to the point that the node can iterate forward and backward depending on if the index is in the first half or in the second half. If the index in the first half the list will iterate forward, if the index is in the second half the list will iterate backward from the end of the list. The graph below shows that the DoublyLinkedList exhibit O(N) while for arraylist it shows O(1).



For the remove method the big O for DoublyLinkedList is O(N). This is also the Big O for ArrayList which is also O(N). This is shown in the plot below which show the two display the same growth rate which is O(N). The reason for this is that the DoublyLinkedList has to iterate through the list to remove the item. The arraylist, while not having to iterate through the list, needs to shift the arraylist one left to shrink the arraylist and remove the item. This causes it to have a big O of N.



2. In terms of functionality and performance, the DoublyLinkedList will do better if you are adding the item at the beginning or at the end. This will have a big O of O(1) while for arraylist this will have a big O of O(N). If you are getting a lot of items from the list than you should use arraylist as it has O(1) for accessing random items. For DoublyLinkedList the big O is O(N), as it has to iterate to the index to get the item. For remove both are about the same as the big O is about the same at O(N). LinkedList will take a bit more space with all of the nodes linking the list so if space is a concern an arraylist should be used.
3. The javas linkedlist differ from this DoublyLinkedList in that it can't iterate backward. So the get function in the DoublyLinkedList is probably a little bit faster. The reason is that if the index is in the first half of the list the Doubly will iterate from the head to the index. If the index is in the second half of the list the Doubly will iterate from the end backward toward the index and get the item. For LinkedList the list can only traverse forward and will be slower and less optimized. Other than that the function and performance are really about the same except the linkedlist also will not have the prev nodes that is in the Doubly.
4. If we were to compare the LinkedList and ArrayList as the back structure for the binarySearch the best structure would be LinkedList. This is because the Binary Search retrieve the item from the first item for the min value. This is what LinkedList is made for as it only has a O(1). Also the

insertion is happening at the same Big O of O(N) for arraylist and linkedlist. The implementation of coding will also be easier with a linkedlist as it deals exclusively with get first .

5. I spend about 7 hours on this assignment.