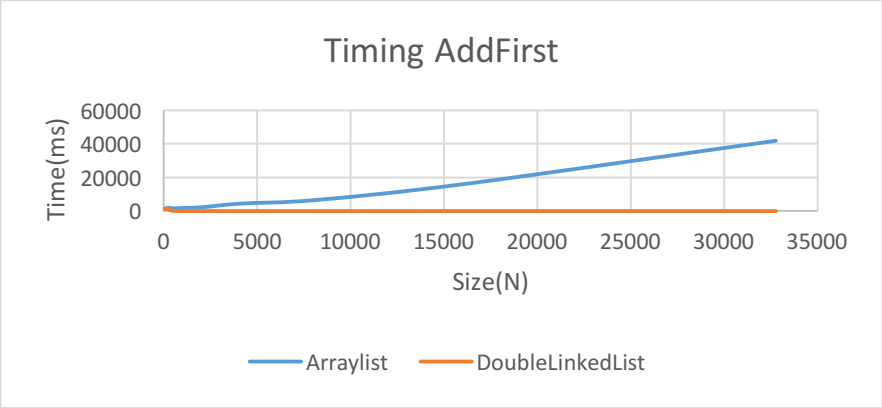
Name:Pingchuan Ma
Uid:u0805309

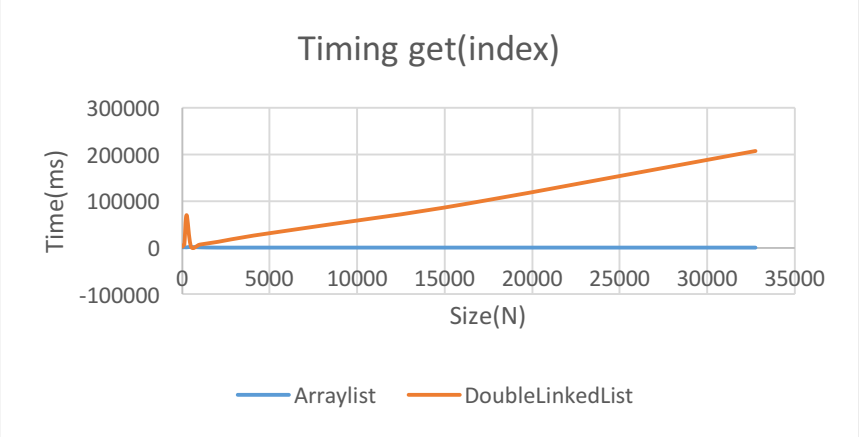
1. Collect and plot running times in order to answer each of the following questions. Note that this is this first assignment that does not specify the exact procedure for creating plots. You must design your own timing experiments that sufficiently analyze the problems. Be sure to explain all plots and answers.

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?

As the chart shows below, the running time of the addFirst method is O(c) as expected.
The running time of add(0, item) for ArrayList is O(N). So the running time of addFirst(item) for DoublyLinkedList compare to add(0, item) for ArrayList always maintain the same.

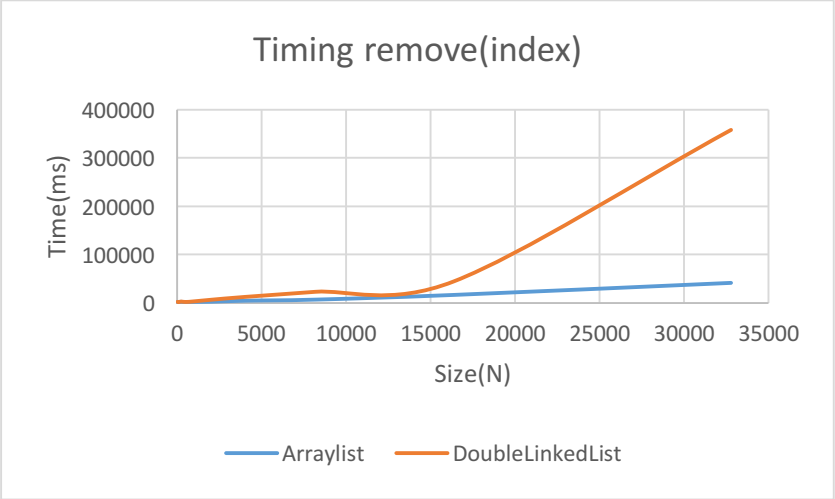


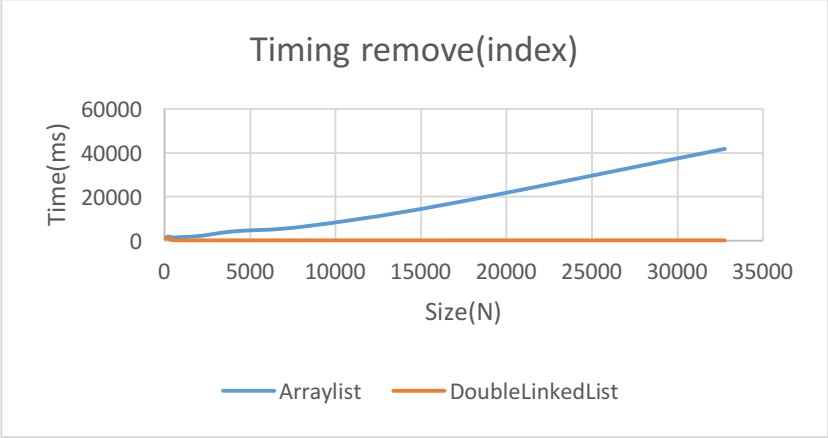
Is the running time of the get method O(N) as expected? How does the running time of get(i) for DoublyLinkedList compare to get(i) for ArrayList?



As the chart shows above, the running time of the get method is O(N) as expected.
The running time of get(i) for ArrayList is O(c). So the running time of get(i) for DoublyLinkedList compare to get(i) for ArrayList always keep increasing as the N grows.

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?





As the first chart shows above, the running time of the remove method is O(N) as expected. Because we randomly remove an element, and we have to traverse to find it first, which is O(N). But if we fixed the index to remove, because DoublyLinkedList do not need to shift the others' index so it is O(c), showed as the second chart above.
The running time of remove (i) for ArrayList is always O(N).

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

ArrayList and DoublyLinkedList have very similar functions, but there are some differences between them.

For search function, Arraylist is much faster, as it uses array data structure implicitly so the get method gives the O(c) performance while DoublyLinkedList requires the traversal through all the elements for searching an element which performace is O(N).

For delete and insert functions, DoublyLinkedList has O(c) performance while ArrayList has O(N) performace. Elements in DoublyLinkedList has two pointers point to the adjacent elements. So when delete or insert new node it only requires change in the pointer location in the two neighbor nodes. While Arraylist has to shift all the elements to fill out the space when

deleting or give space to a new insertion when inserting. If we add and delete element to/at last position, DoublyLinkedList has same performance as ArrayList which is O(c).

Conclusion: DoublyLinkedList element deletion and insertion are faster compared to ArrayList. While ArrayList search is faster than DoublyLinkedList.

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

LinkedList and DoublyLinkedList have very similar functions, their functions include insert remove search clear and so on. But they have much differences as below.

As LinkedList has one direction, DoublyLinkedList has two way directions previous and next. So DoublyLinkedList need more memory than LinkedList for the pointers. In Doublylinkedlist if we know in advance that element to be searched is near the end of the list, then we can traverse the list from the end to save time. But in LinkedList we have to traverse from the beginning.

So if we want to save memory and only update the nodes but not search the elements very often, LinkedList is better. But if memory is not limitation and we want to search something much faster, DoublyLinkedList is better.

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

Big-Oh complexity will not change on add / remove / contains if we use ArrayList. But if we use LinkedList Big-Oh complexity will change.

Using array as the backing data structure, Big-Oh complexity of add / remove / contains is O(N), O(N), O(logN).
Using LinkedList as the backing data structure, Big-Oh complexity of add / remove / contains is O(c), O(c), O(N).
Using ArrayList as the backing data structure, Big-Oh complexity of add / remove / contains is O(N), O(N), O(logN).

5. How many hours did you spend on this assignment?
16