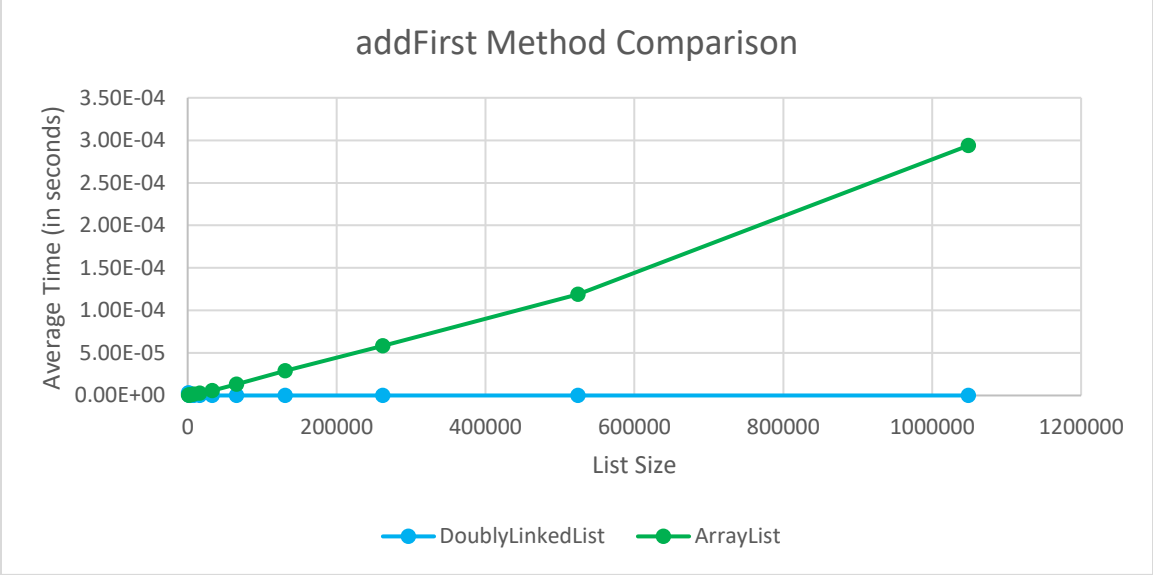
Makenzie Peacock (u0873188)

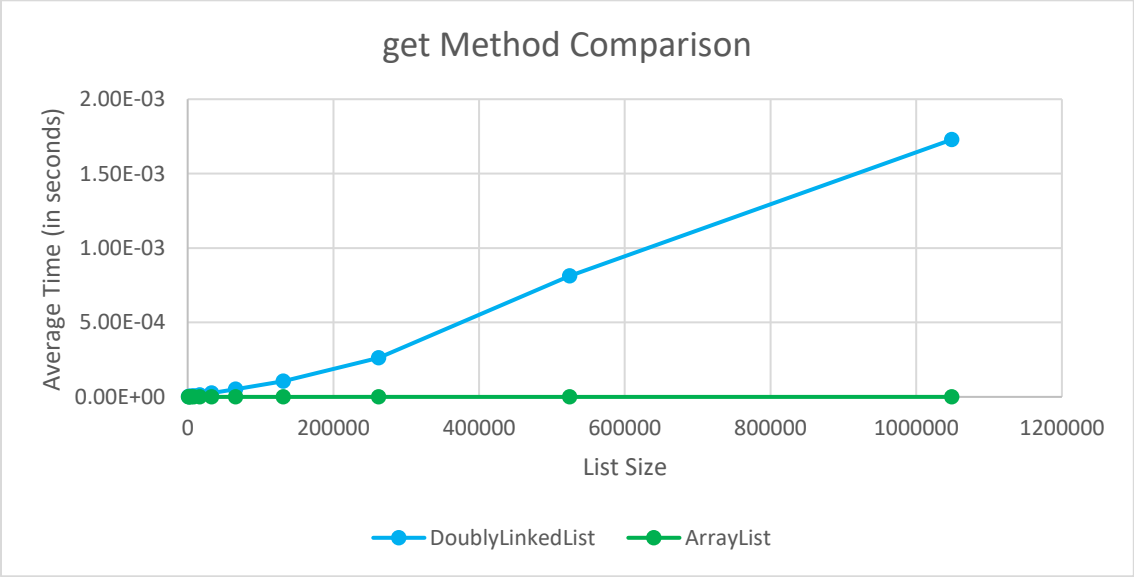
1. **Collect and plot running times in order to answer each of the following questions.**

a) 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?



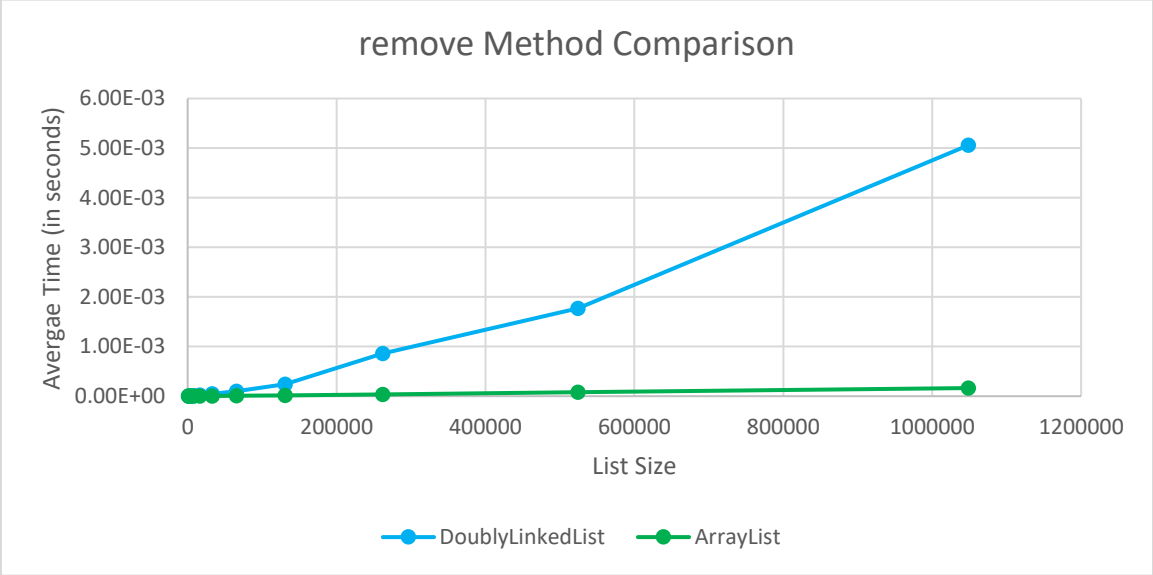
The addfirst method works definitely as expected. It has a significantly faster running time compared to ArrayList's add method.

b) 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 get method shows to have a O(N) complexity as expected. However, it's running time compared to ArrayList's get method is slower.

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 remove method does indeed have a O(N) complexity. It's running time is slower compared to ArrayList though, just like before.

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

Both structures have its benefits and caveats. The DoublyLinkedList is great from adding and removing data and moderate at accessing data. It's vice-versa with an ArrayList however; it's great and performs rapidly when accessing data, but lacks in performance when adding and removing data.

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 are greatly similar in both functionality and performance. The biggest benefit with a DoublyLinkedList, though, is that it can transverse through its data backwards, which may lead to faster running times when accessing data.

4. Compare 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?

With a LinkedList, the Big-Oh complexity would change with the all three methods, but it would be most significant with the add method though. With an ArrayList, the most significant complexity difference would be with the remove method. For the contains method however, both lists would be adequate to back the BinarySearchSet. Since BinarySearchSet's main function is to searching throughout the set, it's a toss-up between the two.

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

About 10 hours.