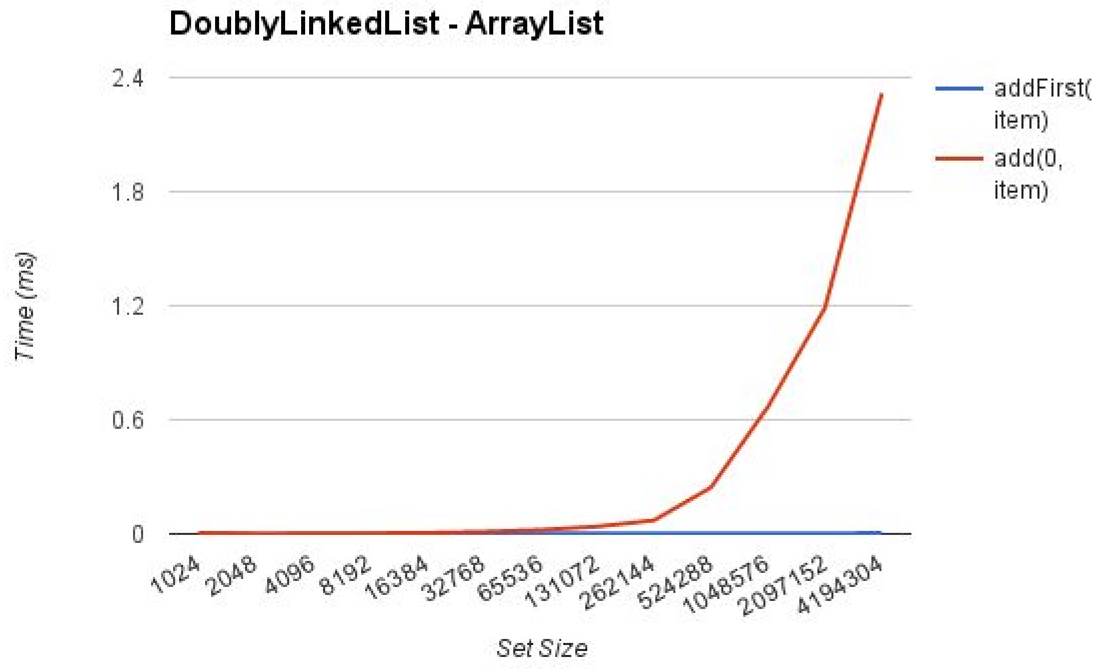
Assignment 6 Analysis

Brayden Wright
u0895942

1. **Collect and plot running times in order to answer each of the following questions. Note that this is the 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(1) as expected? How does the running time of addFirst(item) for DoublyLinkedList compare to add(0, item) for ArrayList?

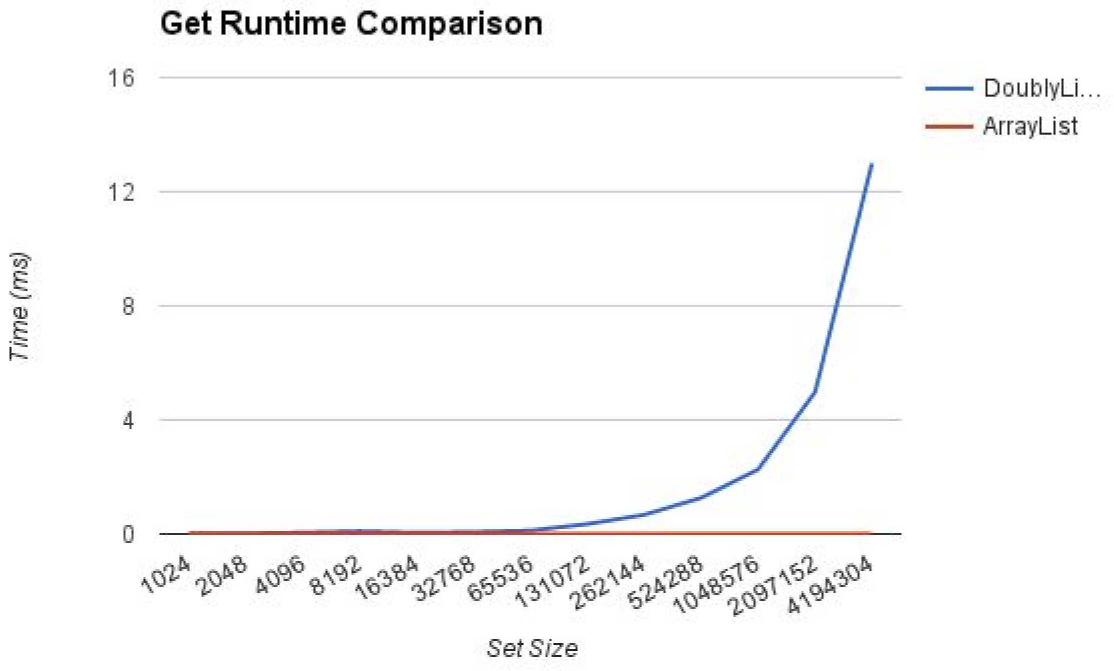


I expected the running time of my addFirst method to be O(1), or constant time. After doing some tests, it appears to be really close to constant (the barely noticeable blue line at the bottom of the graph), though does fluctuate a little bit when looking at the raw data. In any case, it runs better than ArrayList's add(0, item), which appears to have a running time of O(N). The difference is likely because the DoublyLinkedList does not have to shift all the elements after adding to the list, whereas the ArrayList does.

The graph itself appears to show O(N^2), but that's simply because of the growth of the set size is also N^2. The raw data itself clearly shows O(1), or constant, running time.

Brayden Wright
u0895942

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?

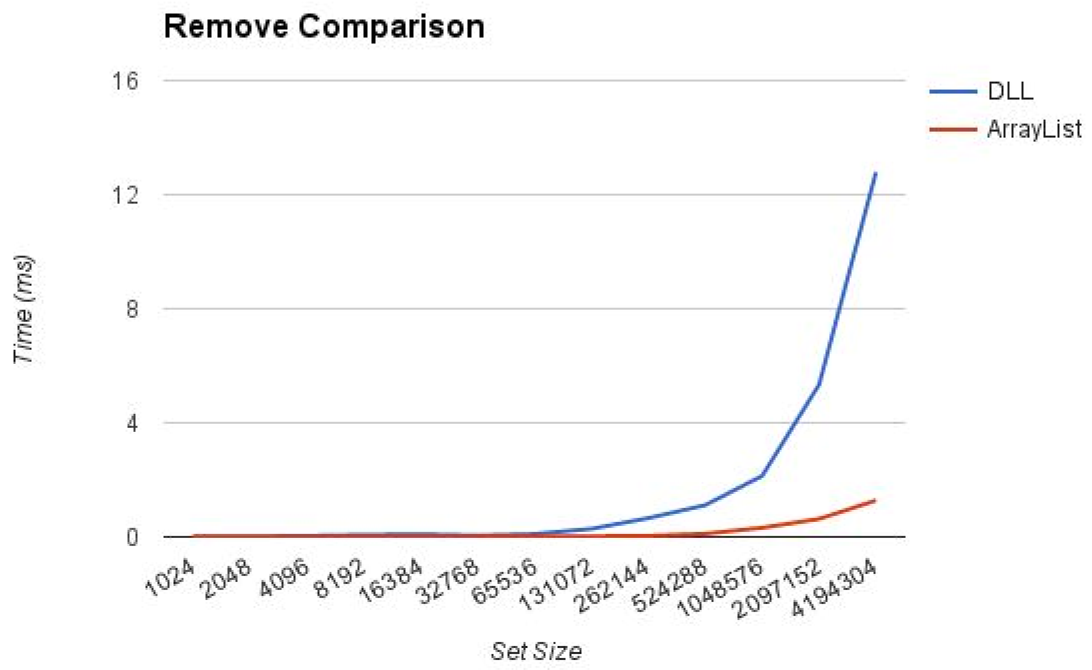


The running time of my DoublyLinkedList appears to be O(N), as was expected. The graph looks a N^2 again because of the growth of the set size, which again grows at a rate of N^2. The raw data again clearly shows a growth rate of N. Oddly, the get method for Java's ArrayList ended up being constant for my timing test. That could be chalked up to my timing test not being set up in the best way for testing the ArrayList method, even though it's set up the same way for the DoublyLinkedList test.

(Analysis continues on next page; graph wouldn't fit here.)

Brayden Wright
u0895942

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 running time of my DoublyLinkedList's remove method is about O(N) as expected. Refer to the last two answers for why the graph appears to show something different. Mostly this running time is due to having to move over to the index point. Both removes are have O(N) running time, though my DoublyLinkedList's implementation appears to run a little slower.

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

In general, DoublyLinkedList would probably the best list structure to use if one has a large amount of data that needs to be added to the beginning or end of a list. Those two operations, adding to the beginning or end, are constant time operations, thus performing better than doing the same thing with an ArrayList. If one needs to add / get / remove a large amount of data elsewhere in the set, an ArrayList is likely to be more beneficial.

Brayden Wright
u0895942

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

Both should be about equal, given both are doubly linked lists. Java's is likely to perform better though, given there are likely some optimizations throughout their implementation that are not in my DoublyLinkedList.

If we were to assume Java's LinkedList was a singly linked list, then the doubly linked list would generally be better. Mostly due to the ability to traverse through the data forwards or backwards, allowing for the ability to optimize the structure if you can predict which part of the structure input data will be stored in, e.g. near the head or tail specifically.

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

Yes, the complexity of the add / remove / contains methods would change. I predict that they would change for the worse. From the tests I did during this assignment, an ArrayList would give better performance when accessing inner data. Since BinarySearchSet relies on accessing the inner data much more than the data near the endpoints, an ArrayList would likely be better. At worst though, they would end up performing about equally.

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

I'm pretty sure the programming part of this assignment took about five hours. Making the graphs and completing this analysis took another hour or two.