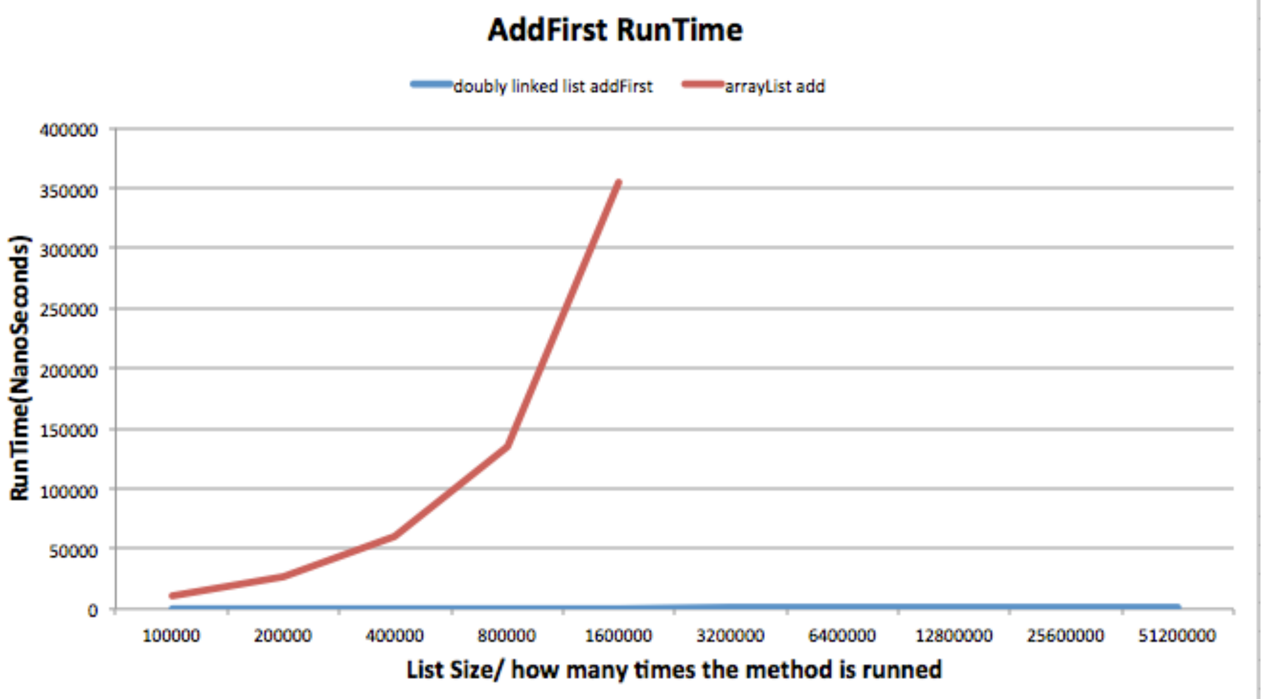
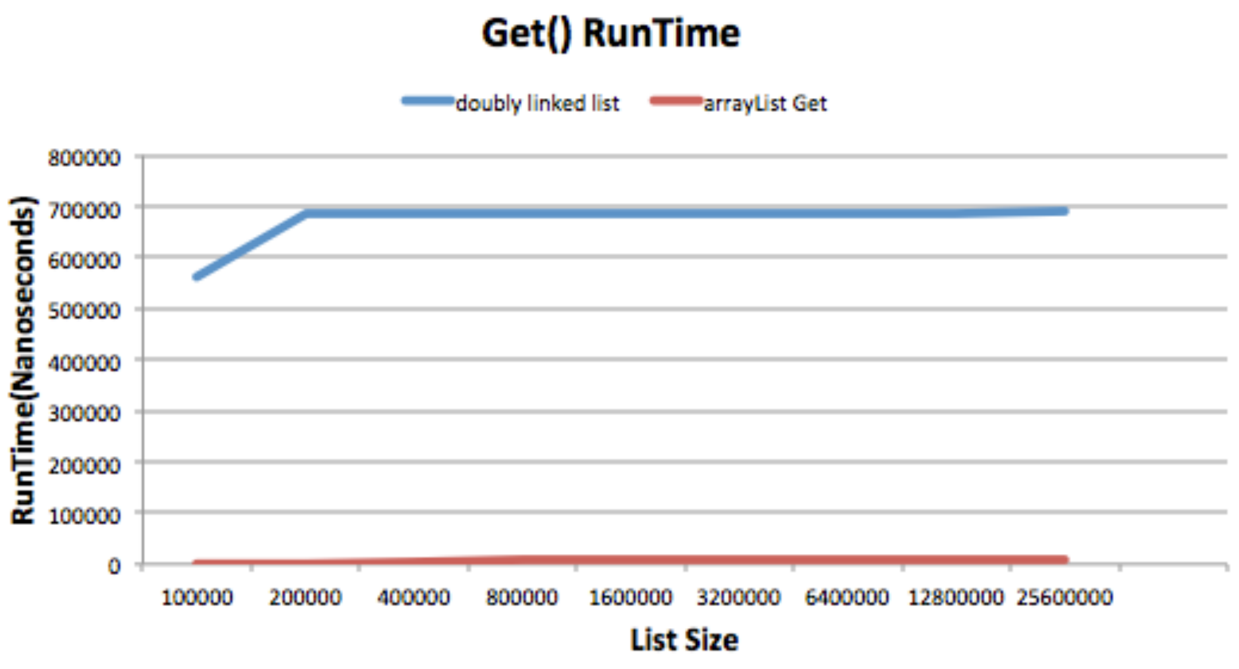
Ching-Yuan Chang u0914005

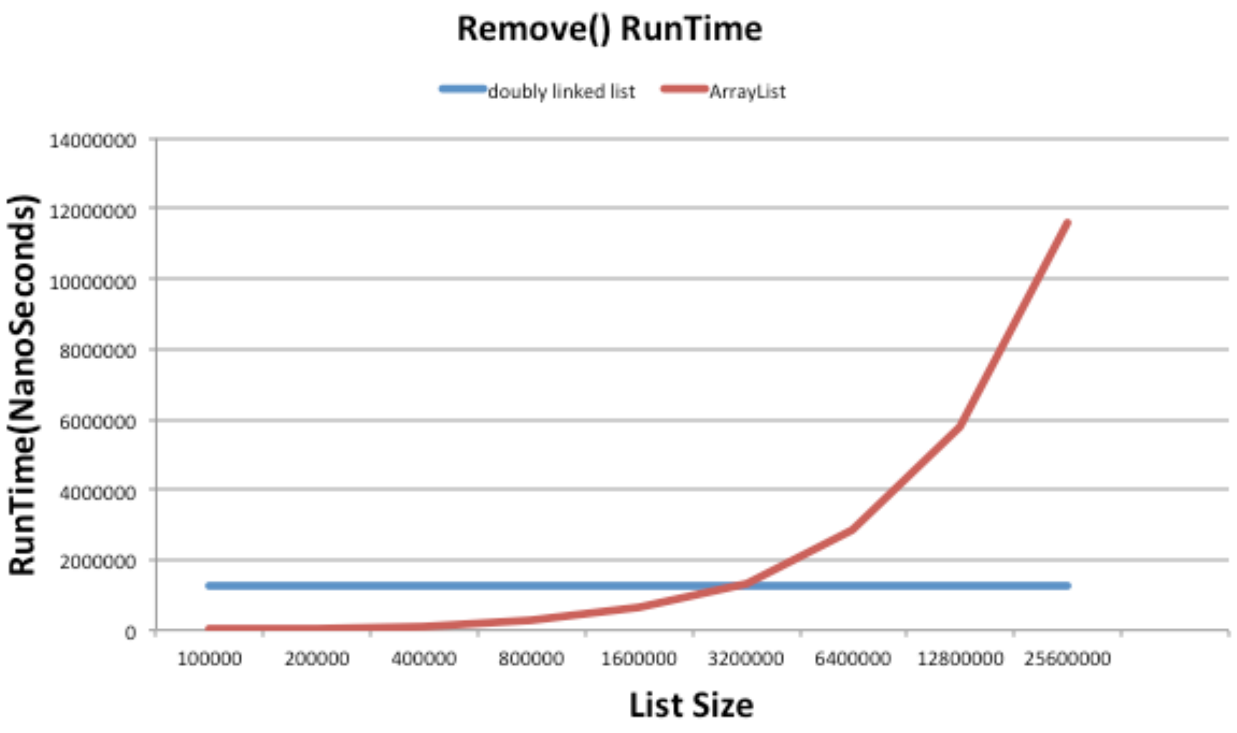
Assignment06 Analysis Document
1.



To time my add first method I run the addFirst method both for doubly link list and Arraylist add(0, element). The size of the element that I added to the list is starting from 100000 and I doubled it 10 times until it reaches 51200000. My doubly linked list run time is just as I have expected which is O(C) constant time. And the run time for Array list is also as expected which is O(n^2). The red line is being cut off at size 1600000 because it is taking forever to run, I actually waited for almost 5 minutes. But you can still see that the run time is O(N^2). Doubly linked list Definitely runs faster, because to add things in array list you would have to move the rest of the elements in array back to create the space in the front, which will take a lot longer.



For the get method, I also use the same size of the array, but this time I created the array before hand, and I run the get method both for doubly linked list and array list that gets the element around middle (which I did size/2). This gave me the graph above. As you can see the Big-O for array list Get is O(N) (you will see it if the graph does not contains the run time for doubly linked list), which is as expected. Also the Big-O run time for doubly linked list is O(N) which is also as expected. Even thought both Big-O are O(N) but array list runs faster and closer to constant time.



For timing the remove method I did the exact same thing as timing our get method. I always remove the item at index size/2 and I created different size of array for it to remove. As you can see from the graph above, the remove run time for array list is O(N^2) which is what I was expecting because when you remove some element you have to move the rest of the element forward so you don't have gap in your array. And the run time for doubly linked list is O(C) which is not what I was expecting. It runs faster then I expected. Maybe it is because I check which half of the list the index is in to decide if we want to start from the head or tail to iterate through. Other then that, doubly link list definitely runs faster and is more efficient then array list.

2. In general, doubly link list runs faster when the size of the list gets really big except for the get method. Array list's get method runs much faster then doubly link list's get method. I think the functionality are similar between the two, its just the implementation is different. Doubly linked list uses nodes, which is better and more efficient when adding and removing things, and Array list uses array and when adding and removing things it have to shift elements back and forth, which will take more time then doubly linked list, but it is easier to implement compare to doubly linked list.

3. Java's link list will take up less memory space because it only uses one link, so you don't have to store the previous data. But in this case some of the run time will be slower because unlike doubly link list it cannot iterate backwards. So for example, to implement the get method I if the index closer the tail, I would start from the tail and iterate through to get the element at the index, which will save a lot of time when the list is really big. But for Java's link list, It doesn't iterate backwards, so in this case you would have to iterate through almost everything because the index is close to tail. it is the same for add and remove methods as well.

4. Since the add/remove/contains method is are all passing in elements instead of index, I think using doubly link list or Array list doubly link list will still run faster then Array list. Even though you will have to look into the nodes and comparing the elements, but the adding and removing process are still faster then array list because you don't have to shift every object if you are adding or removing element from index 0. But for the contains method, I think the Big-O run time will be the same for both doubly link list and array list. Because you would have to iterate down the list, but you don't have to do anything to the list, which is the most significant time difference of doubly link list and array list.

5. For this assignment, I approximately spent 6 to 7 hours on it.