Method	LinkedList Runtime	Explanation	ArrayList Runtime	Explanation	Compare which one is better
add(T element)	O(n)	In my add method, I use a while loop. In this loop, I loop through all of the nodes in my linkedlist and find the last node and set the element to the next node of the last node. Because it goes through all of the nodes to the end, so the complexity is O(n).	O(n)	Add an element at the end of the array, the complexity is O(1) In the worst case, when the size = array.length, the complexity is O(n).	ArrayList is more efficient, because on average, the complexity of ArrayList add is O(1)
T getMax(int n)	O(n)	In my getMax method, I have a while loop to loop through all of the nodes in my linked list and compare the data in each node to the first node. If the next element is bigger than the max one, then change the element to the maximum one. Because it goes through all of the nodes to the end, so the complexity is O(n).	O(n)	In my getMax method, I have a while loop to loop through all of the nodes in my linked list and compare the data in each node to the first node. If the next element is bigger than the max one, then change the element to the maximum one. Because it goes through all of the nodes to the end, so the complexity is O(n).	They are equally efficient.

intersect(List<T> otherList)	O(n+m)	Loop through all of the nodes in both my linked list and other list so its time complexity is O(n+m).	O(n^2)	Need to loop through one array, then loop through another array to check if the element is present in the current list, The complexity is O(n^2)	The linked list is more efficient, because O(n+m) < O(n^2)
reverse()	O(n)	Loop through from the first node to last node and change the direction of the pointer. So the time complexity is O(n).	O(n)	Loop through half of the array, and switch the element with the other half. The complexity is O(n)	They are equally efficient.