Debrian Figueroa

Commentary

The computational complexity of insertEnd was O(1) because the end of the list was recorded using a tail pointer. For the insert method it was O(n) where n is the value of the index passed in because that position needed to be found before inserting the new node. The methods for editing and deleting are both O(n) as well for the same reasons as the insert method. The print method has a complexity of O(n) where n is the size of the list because it must iterate throughout the whole list in order to print out every value. The search method has a complexity of O(n) where n is the size of the array because worst case the value that is being searched for is at the end of the list, or it might not even exist.

Linked lists are good for implementing line editors because insertions and new lines are very common. With is a linked list it is quite simple to insert items and it does not require the shifting of values, as would be the case in an array implementation. Although with a Linked list some efficiency is lost in methods like editing a line using an index, which could be done in O(1) time with an array. The main trade off between choosing a Linked list over an array for a line editor is that line insertions are more likely to occur than edits will, and doing an inserting in an array is way more inefficient as all values after the insertion need to be shifted, and sometimes extra memory allocated. Meanwhile, there is a smaller disadvantage with a linked list in terms as of editing since all that needs to be done iterate until the needed index in this case.

I learned that when working with linked lists it is important to consider what pointer has what value because it is very easy to lose information or misplace it with a small mistake. Also, I learned how to read multiple lines of input and convert that into useful information that can parsed easily. If I had to start over I would probably test my implementation of a method when I create it and work out any issues that may arise before moving on to the next method, instead of writing all the code at once, and then trying to debug.