Nicholas Miller

COP 3530

**Complexity**:

**Insertion**:

My linked list has a complexity of O(n) because in the worst case all items in the list need to be accessed at least once to obtain the position of insertion. The average case will have a complexity of Ø(n/2). My line editor calls linked list’s insertion function, so it has a O(n) worst case, and a Ø(n/2) average case.

**Insertion at end**:

My linked list has a complexity of O(1) because only the tail needs to be accessed.

My line editor calls linked list’s insertion, so it has a O(1) worst case.

**Deletion**:

My linked list has a complexity of O(n) because in the worst case all items in the list need to be accessed at least once to obtain the position of deletion. The average case will have a complexity of Ø(n/2). My line editor calls linked list’s deletion function, so it has a O(n) worst case, and a Ø(n/2) average case.

**Print**:

My linked list and line editor have a complexity of O(n) because at most all items of the list need to be accessed at least once to print their data.

**Edit**:

My line editor has a complexity of O(n) because in the worst case all items in the list need to be accessed at least once to obtain the position to edit. The average case will have a complexity of Ø(n/2).

**Search**:

My line editor has a complexity of O(n) because in the worst case all items in the list need to be accessed at least once to determine if they contain the input string.

**Thoughts**:

I believe that using a linked list was probably best way to implement a line editor. The benefits of using an array-based list is that accessing elements is constant time, but only my edit function requires I find the position. Because I used a tail node, insertions at the end of the list occur in constant time which is another benefit of an array-based list. Inserting into the list is the same for both implementations so there isn’t a difference in complexity. Finally, my print and search function need to go through the entire list no matter what, so the complexity doesn’t change between implementations.

I learned a lot about the inner workings of a linked list and its functions. When I first attempted this, I only used a head node, but soon after added a tail node to increase the speed of insertions at the end of a list. I had never made a linked list with a tail node so that was really fun. I think If I were to start over I may use an iterator to point to where I need to insert or delete. Additionally, I could use new shared and unique pointers to automatically delete memory instead of using the delete function myself. In, all I had a fun time programing this project and I believe it helped me become a better programmer.