

Computational Complexities:

`void LinkedList::add(std::string text)`

$O(1)$ as you just add to the end of the list.

`void LinkedList::add(std::string text, int index)`

$O(n)$ since the maximum amount of points in the list you could check is the length of the list.

`void LinkedList::edit(std::string text, int index)`

$O(n)$ for the same reason as the previous method because you iterate through the list until you find the desired index. The maximum index is the length of the list.

`void LinkedList::deleteIndex(int index)`

$O(n)$ because you iterate through the list until you find the desired index.

`void LinkedList::search(std::string text)`

$O(n)$ because you iterate through the entire list looking for the given text.

`void LinkedList::print()`

$O(n)$ because you iterate through the entire list while printing each value.

`void lineEditor()`

$O(n^2)$ because the maximum amount of calls you would make depends when "quit" is entered, and each line of input calls one of the above methods. Let n = the amount of lines of input entered, and let n also equal the amount of computations you make when you call a method. Since the maximum complexity from the above methods is n , the maximum complexity of the `lineEditor()` is $n*n$. Thus, the method is $O(n^2)$.

I believe linked lists are more efficient for a line editor in some but not all cases. It is fast to add to the end, print, and search through a linked list. This involves either inserting at the back for the former instance and a simple iteration for the two later instances. However, inserting within the list and deleting values would be far easier with a vector. The pre-defined functions within the vector would allow this to be done with a few quick statements while a linked list requires iterations that check for multiple different cases. A major disadvantage of using a linked list is that there is no random access of data. This means that one would have to iterate through the list every time a piece of data is needed.

In this assignment I learned about the application of the intricate details that go into a linked list. I had to account for many different cases involved with certain methods. For example, there could not just be one general algorithm for the add method, as the list could have been empty. I also saw the application of linked lists for data storage and analysis during the implementation of the line editor.

If I had to start over I would try to define a general linked list. The linked list I implemented is designed for the line editor as it takes in 1 as its first index not 0. If I designed it in a more general way, I could use the linked list for any other future methods that would need it.