Britton Chesley

COP3530: Data Structures Programming Assignment 1

The following is a commentary about the computational complexity of each method in the first programming assignment for COP3530:

List(): Since this is a constructor it has O(1) time.

~List(): Since this is destructor parses through each node and deletes each one, this destructor has O(N) complexity , where N is the number of nodes

getSize(): just returns the size of the list, has O(1) complexity

insertEnd(): Since my linked list implemented a tail pointer, the complexity of this method is constant (O(1))

insert(): This method has to parse to the given index to add a node at the correct spot, so the worse case complexity for this method is O(N), where N is the number of nodes

delete_index(): By similar logic to the insert() method, this method will have a computational complexity of O(N)

edit(): Have to parse the linked list, leading to a worse case time complexity of O(N)

print(): Has to parse the entire linked list, leading to O(N) time complexity

search(): Could parse the entire linked list all the way to the end of the list, which is O(N) complexity

the determine_command(), determine_digits(), and determine_text() functions that parse the user input are all O(N) worse case, where N is the length of the user input string.

The main while loop's conditional complexity depends on the user inputted command. Since the user input has to be split into its constituent parts, using determine_command(), determine_digits(), and determine_text(), the complexity for determining the full user command is (N), where N is the length of the input string. Then, regardless of the command given, the worse case complexity for a command is O(M), where M is the current size of the linked list. This yields a total computational complexity of O(M*N).

Thoughts on using linked lists for implementing an editor:

Since most of the methods have O(N) complexity for a linked list, a linked list does a decent job for implementing a line editor. It may be useful to have other characteristics for the line editor embedded in the linked list, like page number or current character number. A better implementation may be found with some sort of binary tree, which has O(log n) complexity for searching. Insertion/deletion of nodes may end up being O(N) functions, but that's the same complexity as any of the linked list functions.

What did you learn from this assignment and what would you do differently if you had to start over?

Honestly, I knew how to implement linked lists before this class, so nothing was too difficult to code. I learned how to write better code for linked lists. If I had to do this project over again, I would parse the user input more efficiently, and make only one function that would return a vector of strings containing all necessary information to call the correct function.