Aaron Leopold

COP3503

Professor Cheryl Resch

21 September 2018

Computational Complexity of Methods/Commands

InsertEnd: Inserting at the end of the linked list has a time complexity of O(1) (constant time). Because I implemented a Linked List that contains a reference to the tail of the list (the last node, at the end), the process of appending a node is simply rearranging pointers and reassigning the variable tail to the new node.

Insert: Inserting at a specific "index" of the linked list has a time complexity of O(n). While the process of inserting the new node is actually time complexity O(1), finding the correct index for the node to be placed involves looping through the list from the head up until the correct index. This process has a time complexity of O(n) since, in this instance the worst case scenario being calling the insert at the index n (where n is the size of the list), you potentially have to loop through each node in the list to get to the proper index.

Remove: Similar to inserting, removing an element at a specified index of the linked list will have a complexity of O(n) since the worst case scenario would be to loop through the entire list up until the last node in order to delete it. However, to counter this worst case scenario I implemented an if statement before the loop begins iterating through the list that checks whether or not the index passed in is equivalent to the count variable that keeps track of how many nodes in the list there are. If the passed in index is equivalent to the count, then instead of iterating through the whole list I can simply rearrange/reallocate the appropriate pointers using the tail reference in the linked list class.

Edit: This function has a time complexity of O(n). When the user wants to edit a line in the list, they provide a line number (index) of the line to be changed. The function will call a linked list member function (list.at(index)) in order to get the node at the desired index. This helper function iterates through the list until the index has been reached, and returns the node. At the worst case, the user will request the second to the last node in the list, which means the program would have to iterate through the entire list up to that node, and then returning it. In the edit function, I added a condition that checks whether or not the index passed in would be the last in the list (if it is equivalent to the number of nodes minus one), and if it is I simply return the tail rather than iterating through the list.

Search: This function will have a time complexity of O(n). While it is possible to find a line of text that matches to search parameters on the first node, this function checks for all possible matches within the list. There is no explicit break statement for this function, and while there is a check in the beginning of the function for an empty list that would skip the loop and return, each node will otherwise be tested for matching data, and therefore the time complexity for this function will always be O(n).

Print: This function has a time complexity of O(n), and it will always have this time complexity. The function only loops through the entirety of the list, starting from the head, and accesses and prints the values held in each node. Since the goal is to print every single data from each node we will be iterating through each node in the list. Therefore, as stated previously, the time complexity will always be O(n).

Thoughts on Linked List Implementation of Line Editor

I think using a linked list for implementing a line editor is an interesting yet useful implementation. While an array based implementation would allow the user quicker access to previous lines, I find that when I use a text editor (for writing papers) I rarely need to travel back to previous lines unless I am correcting small errors. More often than not, I will catch an error before the start of a new line, and simply fix it beforehand. Obviously the text editor implemented for this assignment is much more basic than even a simple text editor like mousepad, so the linked list implementation is not impractical. However, when I consider more robust editors, I can see the advantages of an array based implementation. In a program that has even a basic graphical user interface, each character the user inputs will be shown on the screen, which means for each user input the digital page will be redrawn. Iterating through the document and displaying each line will no doubt be quicker when the program is implemented using an array rather than using a linked list. Appending lines to the end of the document would be quicker than the array implementation on average, since eventually the array will need be resized in order to store more lines of text.

What I Learned / What I'd Do Differently

This assigned helped in understanding the situations in which certain data structures might be useful. I was forced to look at the logic I made to solve the task at hand, and analyze the time complexity of each, which is something I don't normally do yet I've learned can be a very good practice. In the past I know I have occasionally drafted a solution to a programmatic problem without considering how efficient it is, I had a "if it works don't touch it" mentality.

In terms of what I would do differently in approaching an assignment like this in the future, I don't think I would necessarily do anything differently. I created a structure for a document that holds the linked list powering the editor, and I had a very user-friendly implementation of a linked list. My first take at the project I implemented the linked list without thinking of the user/other programmer too much, but I decided to start over and try and make the list a more approachable data structure. I simplified public function calls in regards to their names and parameters, and had those simple functions call the more complex private functions so that the user does not have to think as much when they want the list to perform some specific task. In the future I would like to approach this project again in a more complex and robust way, making it an actual GUI application rather than command-line. Creating my own simple text editor would be quite the challenge, but this has certainly given me the proper footing.