Alex Liou

Commentary on Programming Assignment 1

Professor Cheryl Resch

1. What is the computational complexity of the methods in the implementation?

   The computational complexity would depend on how many times the linked lists is being looped through in order to fulfill its goal. How many times and how far along the list the search is being done.

   insertEnd: This method is O(n). Because we do not have a default pointer for the end of the lists, we have to traverse through the entire list to reach the end in order to append the next line.

   insertAt: Because the position we insert the line can be anywhere in the list, we have to assume worst case scenario to determine O-time complexity. In this case, if we must insert at the end, then for the same reasons listed above for the insertEnd method, the time complexity is O(n).

   deleteLine: Because the line selected can be anywhere in the list, we have to assume worst case scenario to determine O-time complexity. Similar to the insertAt method, if we must delete the last line, then we must traverse the entirety of the list, hence O(n).

   Edit: The edit method is O(n) for the very same reasons as listed before. If last line is to be edited, the method traverses through the entire list.

   Print: The print method is simply the Edit method but missing the lines necessary to edit the targeted line. Hence, it's simply O(n) as it's traversing through the entire list, printing each string along the way.

Search: O(n). Worst case scenario. Search at the end = go through entire list to reach end.

2. Your thoughts on the use of linked lists for implementing a line editor. What are the advantages and disadvantages?

   The linked list is the obvious choice for implementing a line editor. As long as the list is doubly linked, it is very easy to move forward and back along the list and insert and delete new lines without having to shift arrays around to account for the empty cells. However, the downside is that it requires looping from the very beginning to reach a specific line. More efficient algorithms such as binary search or quick search to find the target node could make the node faster, but it'll still lose in access speeds compared to an array where as long as you have the index, you can access the cell in O(1) time.

3. What did you learn from this assignment and how would you do it differently if you had to start over?

   I learned about the use of Linked Lists and their advantages and disadvantages over arrays. To be honest, I was already aware about their qualities due to Programming & Fundamentals 2 but the professor didn't really teach that class well so I'm fine relearning and honing my experience with these data structures. If I had to start over, I'd probably do it the same for lack of effort and better things to do. If I had to improve it, I'd make it doubly linked and have a Node pointer to keep track of the end of the list so that worst case scenario can be O(n/2) which is still O(n) but at least it'll be fast in most test cases. I'd also experiment on finding more efficient ways of searching through the linked list, such as giving each Node an index number and using binary search to find the specific node, cutting down on time complexity.