Data Structures and Algorithms: Programming Assignment 1, Commentary

Nancy Newlin

A) What is the computational complexity of the methods in the implementation?

print: O(n)
calculateSize: O(n)
insertNode: best = O(1), worst = O(n) // best if inserting at head
deleteNode: best = O(1), worst = O(n) // best if deleting head
insertEnd: best = O(1), worst = O(n) // best if first node of list
edit: best = O(1), worst = O(n) // best if editing first node
search: O(n)
findWord: O(n)

Overall Complexity score: 295.56

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

The practicality of linked lists for a line editor varies with its intended use. Most of the methods were hindered by the need to loop through the whole list to operate on a desired node. Such methods were search, insert, and delete. Although, deleting or inserting on the head node is O(1).

However, in this scenario linked lists are simpler than a stack for deleting/editing the ith element. While a stack or queue would have constant complexity for adding or deleting to an end of the list, they would not have an advantage for other access or edits. Editing and deleting with a linked list is much more efficient than a stack or queue because replacing a node is O(1). With a different data type (array based), the searching algorithm could have had a better time complexity if binary search (O(log(n)).

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

Throughout the project, I saw places where having an ongoing list iterator object would have lowered the computational complexity. For example, the node for the next insertEnd could be tracked by an iterator. This would result in had O(1) complexity instead of O(n) caused by having to loop through the list each time to find the last node. I could also implement a tail node, which would make insertEnd O(1).

Finally, I would also like to add a more sophisticated line input parser. I feel that my current version would not work well in more general cases, or situations where there are more methods, or more items to read in.