Andrew Sowinski
9590-1117
September 21, 2018
Programming Assignment 1

Computational Complexity:

Class node: O(1): Simply a class definition with constant time

getPhrase(): O(1): A constant time method that uses if-statements for input parsing

insertEnd(): O(1): Because I keep track of the last node of the list, we do not have to iterate through the entire list to get to the end, resulting in constant time.

inserter(): O(n): This method has one while-loop that iterates through the list, resulting in linear growth as the list grows. This does include a call to getPhrase(), but that does not change to complexity because getPhrase() is linear.

printer(): O(n): Has to iterate through the entire list to print it, so the time is linear.

deleter(): O(n): Has one while loop and calls a constant time method but maintains O(n) time because it only needs to iterate through the list once.

searcher(): O(n): Also only has to go through the list a single time, so the time complexity is linear.

editer(): O(n): As most of these methods are, this one is linear to iterate through the list

looper(): O(n): This method will execute once for each input that the user gives, resulting in linear complexity.

main(): O(n): While this method only executes once (so is constant), the called looper() method is time complexity O(n), resulting in an O(n) time complexity for this method.

Analysis:

At first glance Linked lists seem like a bad choice because it is so hard to navigate them, but they have several definite advantages. They are extremely scalable and will only take up as much memory as they need, unlike more other data storage structures like an array. They also are great for adding or removing elements within the list, because a linked list can collapse or expand with O(1) complexity (though finding the right index does take O(n) time). In a text edit application, this is a very important feature.

The main drawback of using linked lists in a text editing situation is that they are not good at all for getting to a certain index. In order to do so, you have to iterate through the entire list and count as you go. This time can be cut down by storing landmark nodes as variables, but this is only applicable to some situations. Overall, using a linked list for this assignment went much more smoothly than I thought, and I will have to keep them in mind during future personal and professional projects.

Reflection:

The main thing that learned from this assignment is that it is important to know your program specs before you begin programming. Because Stepik is a new platform and this assignment relied heavily on platform and test case compatibility, I had to spend lots of time monitoring the Slack to make sure that I was up to date on the latest expectations, which was frustrating but completely understandable. I played it safe by covering as many test cases as I could think of, but I sacrificed efficiency by taking this route. I just hope that I did not forget anything obvious as a result of covering the not-so-obvious. If I were to start this project over, I don’t think I would change anything. It was a fairly easy assignment, and I am satisfied with the result.