Project 1 – Commentary

Gregory DeCanio

COP 3530

TA: Jose Torres

Section: 3541/22600

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

void insertEnd(std::string lineWords);

- O(1). There are no loops. The new node is simply added to the end.

void insertIndex(int lineIndex, std::string lineWords);

- O(n). There is one loop used to iterate to the correct index for insertion.

void deleteIndex(int lineIndex);

- O(n). There is one loop used to iterate to the correct index for deletion.

void editIndex(int lineIndex, std::string lineWords);

- O(n). There is one loop used to iterate to the correct index for editing.

void printDoc();

- O(n). The method must go through one loop an entire "n" times in order to print off each node in the list.

void searchDoc(std::string lineWords);

- O(n^2). The method must go through one loop an entire "n" times in order to find the specified string in each node in the list (and then print it if applicable). In addition, each node's string uses the "find()" function, which has O(n) complexity, making it a loop inside of a loop, and thus O(n^2).

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

Advantages: Linked lists allow data to be dynamically rearranged and edited in a list due to its pointers. It is easy to change the pointer from one node to another, changing the order. This would be a lot harder to implement in an array, where the data is order on where it is in memory. Another advantage is its ability to insert at the beginning or end of the list easily (O(1) time, if a head and tail is present on the list).

Disadvantages: Linked lists are costly when it comes to iterating through them to find specific items, insert items, or delete items. In most cases, the list will have to iterate through the entire list to find a value (unless you want to access by index, which could hypothetically reduce time by using a binary search type method). This can be very costly, especially if the list is large, and if you need to iterate through each node's string specifically to see if the text you are looking for is in that node. Iterators can be used to decrease this complexity, but will not always drastically decrease the complexity.

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

In this assignment I learned a lot about linked lists, creating your own data structure to fit your needs, commenting and reviewing code, and overall code construction. It is important to really think about your code, and how you want to design it, before really making it. I talked about my code design and purpose with a TA beforehand, which proved to be really helpful. Together, we came to the conclusion that an iterator was not necessary for a project like this, which really helped slim down what I had to create, and also allowed me to focus more on how I should structure my data structures in order to implement the required functions. This assignment also showed me the flexibility of creating my own data structures. By using the concepts of existing structures (such as lists) I had a strong understanding of what I needed to create, but I was able to only include what I needed. This project also reminded me the important of commenting my code, and reviewing all portions of the code before tackling a problem. I spent nearly an hour and a half trying to figure out why my code was printing out quotations around the string, only to find that 3 days ago I had purposely inserted quotations in my code. Overall, I learned that it is important to plan out how you want your structure to look, make sure each aspect of the structure has a necessary purpose, and design your structure to your needs.

If I had to do this assignment differently, I would probably manage my data a little better, making more aspects of the structures private. I would also comment better throughout, instead of skipping commenting momentarily because I felt like I was “in a flow.” Furthermore, I would see if I could maybe decrease the complexity of my find command.