Rohan Kharwadkar

COP3530

Professor Resch

9/21/18

Commentary

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

* void addToEnd(string text) O(n)

The addToEnd(string text) method has O(n) time complexity because it iterates to the end of the linked list which has ‘n’ number of nodes and then adds a new node at the end.

* void print() O(n)

The print() method has O(n) time complexity because it iterates to the end of the linked list which has ‘n’ nodes and prints out each node’s value along with the line number.

* void insert(string text, int position) O(n)

The insert(string text, int position) has O(n) time complexity because it iterates to a given position in the linked list, however it may have to iterate all the way to the end of the linked list. The linked list has ‘n’ nodes. Once it has reached the given position it inserts a new node.

* void deleteLine(int position) O(n)

The deleteLine(int position) has O(n) time complexity because it iterates to a given position in the linked list, however it may have to iterate all the way to the end of the linked list. The linked list has ‘n’ nodes. Once it has reached the given position it deletes that node.

* void edit(string text, int position) O(n)

The edit(string text, int position) method has O(n) time complexity because it has to iterate to a certain position in the linked list. It may have to iterate all the way to the end of the linked list. The linked list has ‘n’ nodes. Once it reaches the node given positon it edits the data of the node at that position.

* void search(string text) O(n)

The search(string text) method has O(n) time complexity because it has to iterate through the linked list until it finds the node which has the value that needs to be found. The linked list has ‘n’ nodes. Once it finds the node that needs to be edited the value of the node is set to the input text given by the user.

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

I personally thought that using a linked list was a great way to implement a line editor. The biggest advantage of using a linked list was that there was no maximum capacity to the number of lines that could be added to the line editor. If we had used an array then once we hit the maximum capacity we would have to make a new, larger array. The downside of using a linked list is that we were unable to have constant look up time. Look up time for this line editor was O(n), whereas the look up time for an array is O(1).

Another advantage of using a linked list was the ability to very easily remove a line from the line editor. All I had to do was have the node before the one I wanted to delete point to the node after that one I wanted to delete and then delete the node I wanted to delete. If I had to use an array then deleting from the middle would become a lot more complicated as I would have to move all the elements over one position.

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

This assignment taught me a lot about implementing linked lists, different C++ libraries and designing efficient algorithms. I learned how to properly create a linked list and how to write efficient methods to work with it. I had previously worked with C++’s string library however this was the first time that I found myself trying to figure out how to use its different functions to help me improve my code. In my project, I frequently used the string library’s substring function, find function and stoi function. These helped me clean my user input, search for strings and convert strings to integers. Without, these I would have had to write my own, longer methods to perform these tasks. This assignment also pushed me to write code that was efficient. I would sit down with a pencil and paper and draw a sample linked list and then write my code based on that. I was able to ensure that none of my code had more than O(n) time complexity.

If I could start over then I would try to make my code modular. I found myself reusing code that I had written in one place in different places. This is definitely an indicator that I should have put that code into a method and then just called it multiple times. For example, I used the same piece of code to check if the line number was more than one digit in three of my methods. Another change I would make would be to create a node pointer that points to the tail of the linked list. This would allow me to add to the end of the linked list in constant time, O(1).