COP 3530 Assignment 1 Report

Computational complexity of methods:

handleInsertEnd:

O(1)

handleInsert:

O(n)

handleDelete:

O(n)

handleEdit:

O(n)

handlePrint:

O(n)

handleSearch:

O(n)

Thoughts

I think a link list is a good choice for a line editor because of the efficient distribution of computational complexity. The algorithms in use are easy to use and never surpass O(n). Although writing a line editor with JavaScript and its thousands of libraries would be easier on behalf of all its array prototype functions, they are expensive and slow. Using a linked list for a line editor allows you to insert, delete, and search without having to parse, concatenate, or cut up any arrays or lists. The programming is a little complex, but the methods used in a linked list line editor are extremely efficient and fail proof.

What I learned

After taking the summer off C++ and personally getting more involved with web-development, this project was a great re-introduction to C++'s concepts (especially pointers). The most important skill I learned in this project was how to make a traversable set of data by using a customized struct. I believe the knowledge I've gained from working with nodes will be applicable to many future projects, as ordered data and its inherent relationships are ubiquitous in the world of programming. Learning to point to related data and then reference those pointers will prove to be extremely beneficial in the future.

If I had to start this project all over again. I would make the list class a little more complex and use member functions instead of helper functions floating around arbitrarily in the code. I think this would make my final submission seem more professional, and it would become objectively more scalable and legible to my peers.