Bernard Mingo
9/20/2018
COP 3530

Linked List Line Editor

Thoughts on linked list implemented as a line editor

Personally I am a fan of the linked list data structure. Not only is it simple to interact with, but the reference based data storage makes interacting with the data more straightforward as opposed to the standard array data type. However, as a line editor I feel like this data type may not be completely necessary or at least optimized, there is no explicit reason for each line to need to know the location of their nearby nodes. While using a linked list is not a 'bad' implementation - the same could have been achieved with a vector or a well implemented array.

Computational complexity of the methods in the implementation

As a list data type - most of the functions implemented in the linked list take O(n) time as you must iterate from the head (or tail) to the desired node of the desired value. Such functions include the *PrintAll, PrintForward, RemoveAt, InsertAt, and editAt* functions. However - by keeping a reference to the tail - in constant time one can add nodes to the tail of the list with ease as is the case with the *insertEnd* function.

What did we learn from this implementation?

Though this is not my first time - or most advanced implementation of a linked list - I believe that the linked list is an amazing lesson to learn. Most programmers who come from a java background - tend to not have a very good understanding of memory management as the virtual machine does most of the work for you. Learning how to avoid memory leaking and segmentation is a great practice for a programer and helps you understand which data structures you might want to apply to your task.