605.202: Introduction to Data Structures

Omar Ismail

Lab3 Analysis

Due Date: April 13th, 2021

Dated Turned In: April 15th, 2021 (Covid Vaccine Extension)

Lab3 Analysis

Using Linked Lists to add, multiply, and evaluate polynomials was a tall task at first. My program has 4 main components, user input/output through .txt file system, a Node class, a class for actually manipulating polynomials, and finally, a class for evaluation.

I choose to implement a doubly linked list, which wasn't that hard to initialize. However, it was much more difficult that our past labs. There were many moving parts in this lab and if you got lost anywhere, it seemed like a maze to find where things went wrong. I will say that although I considered other types of linked lists, I think a doubly did exactly what I needed it to. In fact, I probably should have made a singly linked list because of the way I utilized my doubly. I never went backwards in my linked list.

Is a recursion solution a good choice for this lab? Absolutely, it would actually make so much sense. The reason that I picked an iterative solution, rather than recursive, is due to the many moving parts of this lab. I did not feel comfortable enough with recursion to deploy it in this instance. Recursion is also much more efficient in terms of space and run-time, which my program was not too hot on.

Due to the requirements of this project, we were not allowed to use the built-in LinkedLists library (java.util.LinkedLists) which would make our life much easier. With this library, I could delete two of my classes that I made just to initialize nodes and the actual linked list. Without this library, I found that my greatest challenge was reading input from a .txt file and inputting them into my own linked list. The algorithm for the input took a lot of trial and error.

What I Learned

This was the largest (in terms of moving parts) project that I have worked on to date. I biggest struggle was understanding what wasn't working and where things were going bad. This forcibly taught me to debug. It was a life saver. Taking the time to understand what each part of my code did, overall, helped me understand what my errors were.

What I might do differently Next Time

I think my setup for this project was all wrong. The first thing that I did was tackle on the polynomial math right away. If I were to do it over again, I think I would take my time and completely setup the whole project. I should have started with the way I was going to ready my files from the .txt file. It would have given me a clearer idea of how to manipulate my coefficients and exponents right away. I say this because halfway through the project, I found myself confused on to how I was going to feed my numbers through my functions and it made me really frustrated.

Justification for Design Decisions

There are 5 classes in my program. The main entry point outlines the solution of manipulating polynomials and evaluating them without showing the details of the conversion. This allows for a cleaner code.

The Node class is the initialization of the nodes that were going to be used in the LinkedList class. It's really simple.

The LinkedList class is my version of a doubly linked list. As I've mentioned before, although I created a doubly linked list, I think that a singly linked list would probably have been better for my case.

The polyMath class is where all the conversion happens. It adds, subtracts, and multiplies polynomials using linked lists. At the bottom of this class, I have also added a function that gets input (coefficient, powX, powY, powZ) from the input files.

Finally, the EvalArg class which is meant for the output of the polynomials. This allows my main driver code to look cleaner but adding output strings in another class.