COMP 250 Assignment 2 (Fall 2018)

Prepared by Prof. Michael Langer

Posted: Tuesday, Oct. 9.
Due: Tuesday, Oct. 23, 23:59 PM.

General instructions

- The main T.A.s handing this assignment are Aashima, Abhishek and Sayantan. Their office hours and location will be announced on mycourses. Anmoljeet and Anand will also have office hours and will able to answer questions.

- *Do not change any of the starter code that is given to you. Add code only where instructed, namely* in the "**ADD CODE HERE**" block. You may add helper methods to the Polynomial class, but you are **not allowed** to modify any other class except for Task 1.

- You can use whatever package name you like. However, make sure the package name is the first line of the file, so that the grader code can easily remove it. To learn more about packages, see the Java tutorials.

- The starter code includes a tester class that you can run to test if your methods are correct. *Your code will be tested on a more extensive and challenging set of examples.* We encourage you modify this tester code and to share your tester code with other students on the discussion board. Try to identify tricky cases. Do ***not*** hand in your tester code.

- Your code will be tested on valid inputs only.

- You may submit multiple times, e.g. if you realize you made an error after you submit. We will automatically grade the most recent valid submission.

Late assignment policy

- Late assignments will be accepted up to two days late and will be penalized by 10 points per day. If you submit one minute late, this is equivalent to submitting 23 hours and 59 minutes late, etc. So, make sure you are nowhere near that threshold when you submit.
- See the Submission Instructions at the end of this document for more details.

Introduction

In mathematics, a polynomial is an expression constructed from variables and constants, using the operations of addition, subtraction, multiplication, and constant non-negative integer exponents. For example, $5x^3 - 4x + 7$ is a polynomial. In general, a polynomial can be written

$$p = \sum_{i=0}^{m} c\, x^i \; = \; c_0 \; + \; c_1\, x + \; c_2\, x^2 \; + \; \ldots + \; c_m x^m$$

where the coefficients c_i are real numbers and $c_m \neq 0$. As explained below, in this assignment, the coefficients c_i and the variables x will be integers, represented with the Java BigInteger class.

Polynomials are heavily used in computer science and in mathematics and also in many other sciences. They are the basis of many models in physics and chemistry and other natural sciences, as well as in the social sciences such as economics. For example, they are commonly used to approximate functions. An example is a Taylor series expansion of a smooth function. Another example is that any continuous function on a closed interval of the real line can be approximated as closely as one wishes using a polynomial.

Instructions and Starter Code

We will use a singly linked list data structure to represent polynomials and perform basic operations on them, such as addition and multiplication. Essentially, you will write a Polynomial class that builds upon the functionality of a linked list class. The starter code defines four classes which are as follows:

- **Polynomial -** This class defines a polynomial in one variable with positive integer exponents. Most of your work goes into this function. You should use the methods provided with the linked list class to implement the methods of this class. You will notice that the template we provided use Java BigInteger for storing the polynomial coefficient and variable. We intentionally chose BigInteger over floating point (e.g. double) to avoid numerical issues associated with polynomial evaluation.

- **SLinkedList –** This class implements a generic singly linked list. You do not have to implement the functionality of linked list as it is already provided with the starter code. However, you must implement the cloning functionality (Task 1 see below) for performing a deep copy of the list.

- **Term –** This is a simple class that represents a single term of a polynomial. Example: $5x^2$. This class is also provided with the starter code. You do not have to modify this class, but you are required to understand what it does and how.

- **DeepCopy –** This is a small user defined interface for enforcing deep copy functionality. You are not required to understand Java interfaces for completing this assignment as the underlying complexity is handled by the starter code. We will discuss Java interfaces in the lectures a few weeks from now.

Valid polynomial representation:

Any method that outputs a polynomial or modifies an existing polynomial (adding a term or multiplying by a term) must ensure that this polynomial satisfies the following:

- There must be no term in the polynomial with zero coefficient. e.g: term $0x^3$ is ***not*** allowed.
- The exponents in the list must be in ***decreasing*** order
- Exponents can be non-negative integers only e.g. a term x^{-2} is ***not*** allowed.
- There can be at most one term for each exponent.
- A polynomial can be zero, i.e. **p(x) = 0.** In this case, the polynomial must be an empty linked list with no term. i.e. size of the linked list should be zero.

Methods that you need to implement (100 points total)

The methods are listed below. See the starter code for method signatures and more details about what the methods do.

Your implementations must be efficient. For each method below, we indicate the worst case run time using O() notation. This worst case may depend on either the order m of the polynomial as in the definition above, or it may depend on the number n of terms in the polynomial. Note that $n \leq m + 1$.

1. SLinkedList.deepClone (15 points)

Returns a deep copy of the linked list object. It should step through the linked list and make a copy of each term. The runtime of this method should be $O(n)$.

2. Polynomial.addTerm (30 points)

Add a new term to the polynomial. Use the methods provided by the **SLinkedList** class. The runtime of this method should be $O(n)$.

3. Polynomial.add (10 points)

This is a static method that adds two polynomials and returns a new polynomial as result. You may use any of the class methods. Be careful not to modify either of the two polynomials. The runtime of this method should be $O(n_1 + n_2)$ where n_1, n_2 are the number of terms in the two polynomials being added.

4. Polynomial.eval (20 points)

The polynomial object evaluates itself for a given value of x using Horner's method. The variable x is of BigInteger data type, as mentioned earlier. Horner's method greatly speeds up the evaluation for exponents with many terms. It does so by not having to re-compute the x^i fresh for each term. Note that you *should not* use **Term.eval( )** method to evaluate a particular term. That method is provided for you only to help with testing, namely to ensure that your implementation of Horner's method is correct.

A good resource for understanding Horner's method is:
https://www.geeksforgeeks.org/horners-method-polynomial-evaluation/ .
Alternatively, you may use any other resource explaining Horner's method. Note the method described in the above resource works only when a polynomial has all $m + 1$ terms (up to order m). However, your polynomial representation may not have all terms**.** Hence, you must modify your implementation accordingly.

We emphasize that the efficiency of your solution is very important. If a polynomial of order m has all $m + 1$ terms, then normal brute force polynomial evaluation would take time O(m^2), that is, if you were to evaluate each term individually and sum the results. Horner's algorithm is more efficient, namely it would take time O(m). As such, we will evaluate you on polynomials with a very large number of terms that will "stress test" the O( ) efficiency of your **Polynomial.eval** method, as well as its correctness. Your algorithm should take time O(m).

One heads up about terminology: when we discuss the complexity of various operations in this assignment, e.g. on the discussion board, we will use term "order" in two different ways. One way is to refer to the order m the polynomial. Another is to refer to the O( ) complexity, where in general people often say "order" for "big O", that is, people often refer to O(1), O(n), O(n^2) as "order 1, n, and n squared" instead of "big O of 1, big O of n and big O of n squared, respectively.

5. Polynomial.multiplyTerm (15 points)

This is a private helper method used by the multiply function. The polynomial object multiplies each of its terms by an argument term and updates itself. The runtime of this method should be $O(n)$.

6. **Polynomial.multiply (10 points)**

This is a static method that multiply two polynomials and returns a new polynomial as result. Careful not to modify either of the two polynomials. We recommend you use **Polynomial.multiplyTerm, Polynomial.add** and any other helper methods that you need. The runtime of this method should be $O(n_1 n_2)$ where n_1, n_2 are the number of terms in the two polynomials being multiplied.

Submission instructions

- Submit a single zipped file **A2.zip** that contains the following files to the myCourses assignment A1 folder. Include your name and student ID number within the comment at the top of each file.

  - Polynomial.java
  - SLinkedList.java

- The following are invalid submissions:
  - separate java files rather than a zipped directory

  - a rar file (rather than zip)
  - starter code or a class file within your zipped directory
  - code that does not compile

  Xiru (TA) will check for invalid submissions once before the solution deadline and once the day after the solution deadline   He will notify students by email if their submission is invalid. It is also your responsibility to check your email.

- Invalid solutions will receive a grade of 0.   However, you may resubmit a valid solution, but only up to the two days late limit, and you will receive a late penalty.

- If you have any issues that you wish the TAs (graders) to be aware of, please include them in the comment field in mycourses along with your submission. *Otherwise leave the mycourses comment field blank.*