COS 460a Quiz 2

Dastanbek Samatov

April 21, 2020

Abstract

Implement a library for arithmetic operations with Polynomials in C++.

1 Introduction and Overview

The library consists of a class Polynomial with its respective arithmetic methods and other overloaded built-in operators, such as comparison operators, assignment, increment and decrementation operators. Polynomials are stored in a **map**, where key is the power and value is the coefficient of respective member of the Polynomial. For example, a polynomial p

$$p = x^4 + 3x^3 - 2x^2 + 3 \tag{1}$$

is assigned to the following map: [**[0, 3]**, **[1, -2]**, **[2, 3]**, **[3, 1]**]

Overview of the file structure:

```
/
├── doc
│   └── doc.pdf
├── main.cpp
├── PolyOperations.cpp
├── PolyOperations.h
├── test.txt
└── tests.cpp
```

1.1 Constructors

Polynomial class has four types of constructors: **default, constructor from vector, constructor from map and constructor from another instance of the class.**

1.2 Arithmetical Operations

An instance of **Polynomial** class has add, subtract, multiply, divide methods along with several utility functions. Along with these methods, native arithmetic operators (+, -, *, /) are also overloaded for the Polynomial class.

2 Theoretical Background

Algorithm for adding and subtracting Polynomials is rather trivial. Multiplication is slightly more complicated but it also doesn't deserve specific mention. Some other problems I had was zero coefficients after arithmetical operations. So I implemented a methods which finds those coefficients and erases them from the map of the object.

Another objective was to deal with the Null Polynomial which resulted from the arithmetic operations, since in the clean() method I was iterating through the elements of map, it was essential to handle the iterator so that the next node is not empty. Otherwise, the program would give Segmentation Fault.

3 Algorithm Implementation and Development

Following is the Pseudocode on which I based the implentation of division of polynomials. Worst time complexity is O(n^2) and the best is O(1). It receives dividen and divisor as an argument and return a vector of maps with quotient and remainder.

Algorithm 1: Polynomial Long Division Pseudocode

```
Result: returns (q, r)
if degreeD smaller than 0 then
    error
end
q = 0;
while degreeN greater or equal to degreeD do
    d = D (shifted by degreeN-degreeD);
    q(degreeN-degreeD) = N(degreeN)/d(degreeD);
    d = d * q(degreeN - degreeD);
    N = N - d;
end
r = N;
return (q,r);
```

4 Conclusions

The code is well-documneted, with necessary comments and structure. When writing the algorithms I used mainly Wikipedia and GeekForGeeks website for research. Most of the methods of C++, such as assignment, comparison, incrementation operators are overloaded for the convenience of computing polynomials.

5 References

References

[1] $Polynomiallongdivision(n.d.).Retrievedfromhttps://rosettacode.org/wiki/Polynomial_long_division$

[2] $Polynomial.(2020, April 13).Retrievedfromhttps://en.wikipedia.org/wiki/Polynomial$

[3] $Polynomiallongdivision.(2020, January 14).Retrievedfromhttps://en.wikipedia.org/wiki/Polynomial_long_division$