CSE214 Data Structures
Polynomial

YoungMin Kwon

Polynomial

- Representing a polynomial
  - Coefficient array: the i^{th} element has coefficient for x^i term.
    - E.g. $2x^3 + 5x^2 + x + 7$ is represented as `coef[0]=7, coef[1]=1, coef[2]=5, coef[3]=2`
  - Leading 0s in the coefficient array should be trimmed out (from constructors)
    - E.g. $0x^5 + 0x^4 + 2x^3 + 5x^2 + x + 7 \rightarrow 2x^3 + 5x^2 + x + 7$

Polynomial

- Addition
  - $(2x^3 + 5x^2 + x + 7) + (3x^2 + 2) = 2x^3 + 8x^2 + x + 9$
  - [7, 1, 5, 2] + [2, 0, 3] = [9, 1, 8, 2]

- Multiplication
  - $(2x^2 + x + 3) * (3x^2 + 2)$
    $= (6x^4 + 3x^3 + 9x^2) + (4x^2 + 2x + 6)$
    $= 6x^4 + 3x^3 + 13x^2 + 2x + 6$
  - [3, 1, 2] * [2, 0, 3]
    = [0, 0, 9, 3, 6] + [6, 2, 4]
    = [6, 2, 13, 3, 6]

Polynomial

- Long division algorithm
  - To compute quotient and remainder

$$
\begin{array}{r}
x - 10 \\
x^2 - 2x + 1 \overline{)\, x^3 - 12x^2 + 0x - 42} \\
\underline{x^3 - 2x^2 + x} \\
-10x^2 - x - 42 \\
\underline{-10x^2 + 20x - 10} \\
-21x - 32
\end{array}
$$

```java
public class Polynomial {
    private double[] coef;

    public Polynomial(double[] coef) {
        //trim the leading zeros
        int n = coef.length;
        while(n-1 >= 1 && coef[n-1] == 0)
            n--;

        this.coef = new double[n];
        for(int i = 0; i < n; i++)
            this.coef[i] = coef[i];
    }

    public String toString() {
        StringBuilder sb = new StringBuilder();
        for(int i = coef.length - 1; i >= 0; i--) {
            if(i > 0)
                sb.append(String.format("%g*x^%d + ", coef[i], i));
            else
                sb.append(String.format("%g", coef[i]));
        }
        return sb.toString();
    }
```

```java
public static void main(String[] args) {
    Polynomial a = new Polynomial(new double[] {-1, 1});
    System.out.println("a: " + a);

    Polynomial b = new Polynomial(new double[] { 1, 1});
    System.out.println("b: " + b);

    Polynomial c = a.add(b);
    System.out.println("c = (a + b): " + c);

    Polynomial d = a.mul(b);
    System.out.println("d = (a * b): " + d);

    Polynomial e = d.add(c);
    System.out.println("e = (d + c): " + e);

    Polynomial[] f = e.longdiv(a);
    System.out.println("e / a: " + f[0]);
    System.out.println("e % a: " + f[1]);
}
```

```java
public Polynomial add(Polynomial that) {
    double[] c = new double[Math.max(this.coef.length,
                                     that.coef.length)];
    //TODO: implement the rest
    return null;
}

public Polynomial mul(Polynomial that) {
    double[] c = new double[this.coef.length +
                            that.coef.length - 1];
    //TODO: implement the rest
    return null;
}
```

```java
public Polynomial[] longdiv(Polynomial that) {
    //return value: longdiv(...)[0]: quotient,
    //              longdiv(...)[1]: remainder
    double[] quo = new double[this.coef.length - that.coef.length + 1];
    double[] num = new double[this.coef.length];  //numerator, remainder
    double[] den = that.coef;   //denominator
    int dd = den.length - 1;    //degree of denominator

    //copy this.coef to num because num will be modified
    for(int i = 0; i < this.coef.length; i++)
        num[i] = this.coef[i];

    //the long division algorithm
    //num -> quo * den + num
    //TODO: implement the rest
    return null;
}
```

```
Result:
a: 1.0x^1 + -1.0
b: 1.0x^1 + 1.0
c = (a + b): 2.0x^1 + 0.0
d = (a * b): 1.0x^2 + 0.0x^1 + -1.0
e = (d + c): 1.0x^2 + 2.0x^1 + -1.0
e / a: 1.0x^1 + 3.0
e % a: 2.0
```