2110211 โครงสร้างข้อมูลเบื้องต้น

Polynomial

สมชาย ประสิทธิ์จูตระกูล

Polynomial

```
public class Polynomial {
  public Polynomial add(Polynomial that) {...}
  public Polynomial multiply(Polynomial that) {...}
  public void addTerm(double coef, int exp) {...}
  public Polynomial diff() {...}
  public double eval(double x) {...}
  public String toString() {...}
  ...
}
```

$$f(x) = \sum_{k=0}^{n} a_k x^k$$

ตัวอย่างการใช้งาน

```
public static void main(String[] a) {
  Polynomial p = new Polynomial();
  p.setTerm(9, 10);
  p.setTerm(3, 2);
  p.setTerm(2, 1);
  p.setTerm(1, 0);
  System.out.println(p);
  Polynomial q = p.add(p);

  System.out.println(q);
  System.out.println(q.diff());
  System.out.println(q.multiply(p));
  System.out.println(p.eval(0.5));
}
```

$p(x) = 9x^{10} + 3x^2 + 2x - 1$

$q(x) = p(x) + p(x)$

$q'(x)$

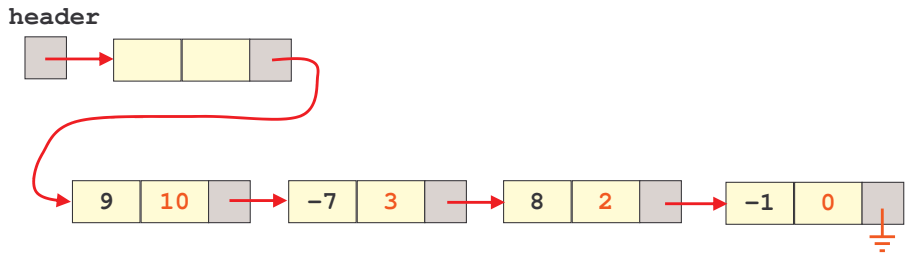
$q(x) \times p(x)$

$p(0.5)$

สร้าง Polynomial ด้วย Linked List มี header

$$f(x) = 9x^{10} - 7x^3 + 8x^2 - 1$$

< (9,10), (-7,3), (8,2), (-1,0) >



เก็บเรียงเลขชี้กำลังจากมากไปน้อย

LinkedPolynomial

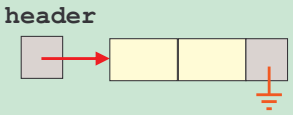
```
public class Polynomial {
  private static class LinkedNode {
    double coef;
    int exp;
    LinkedNode next;

    LinkedNode(double c, int e, LinkedNode next) {
      this.coef = c;
      this.exp = e;
      this.next = next;
    }
  }

  private LinkedNode header = new LinkedNode(0,0,null);

  public Polynomial() {}

  ...
```



LinkedPolynomial : addTerm

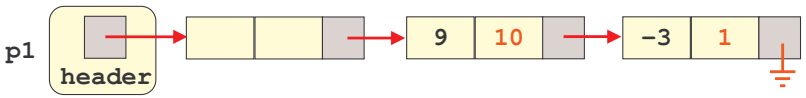
```
public class Polynomial {
  ...
  public void addTerm(double coef, int e) {
    LinkedNode node = header;
    while( node.next != null && e < node.next.exp) {
      node = node.next;
    }
    if (node.next != null && e == node.next.exp) {
      node.next.coef += coef;
    } else {
      node.next = new LinkedNode(coef, e, node.next);
    }
  }
  ...
```

มี n พจน์ ใช้เวลา O(?)

ถ้าไม่มี x^e ก็เพิ่มตัวใหม่ ถ้ามีอยู่แล้ว ก็เพิ่ม coef ด้วยค่าใหม่

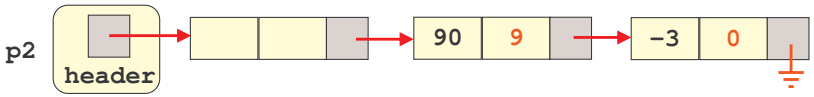
LinkedPolynomial : diff

$$f(x) = 9x^{10} - 3x$$



p2 = p1.diff();

$$f'(x) = 90x^{9} - 3$$



LinkedPolynomial : diff (แบบช้า)

```
public class Polynomial {
  ...
  public Polynomial diff() {
    LinkedNode n1 = this.header.next;
    Polynomial r = new Polynomial();

    while( n1 != null ) {
      r.setTerm(n1.coef * n1.exp, n1.exp - 1);
      n1 = n1.next;
    }
    return r;
  }
```

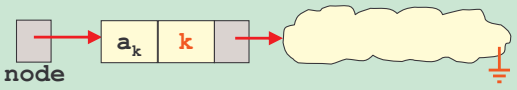
มี n พจน์ ใช้เวลา O(?)

LinkedPolynomial : diff (เร็วแต่ recursive)

```
public class Polynomial {
  ...
  public Polynomial diff() {
    Polynomial resultPoly = new Polynomial();
    resultPoly.header.next = diff(header.next);
    return resultPoly;
  }

  private LinkedNode diff(LinkedNode node) {
    LinkedNode result = null;
    if (node != null) {
      result = new LinkedNode( node.coef * node.exp,
                               node.exp - 1,
                               diff(node.next) );
    }
    return result;
  }
  ...
```

มี n พจน์ ใช้เวลา O(?)



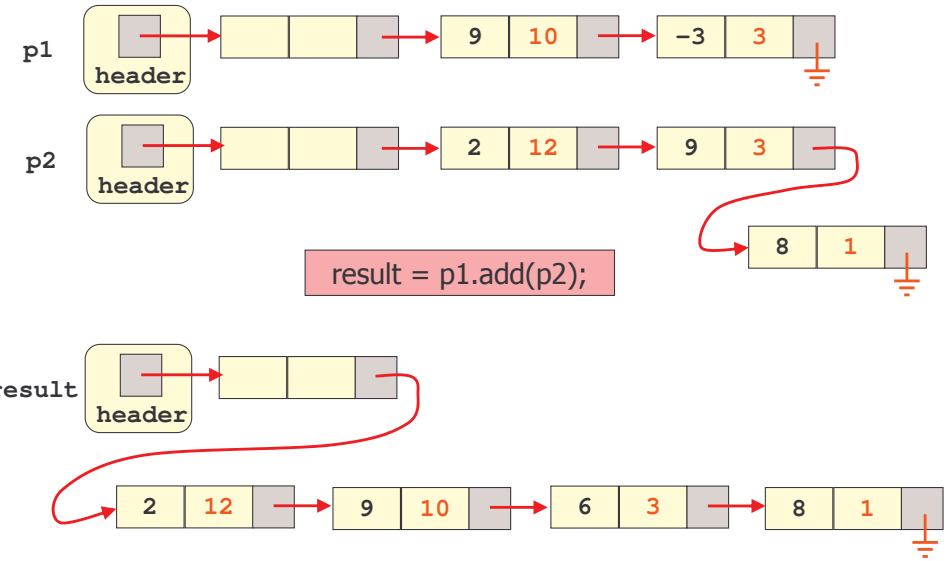
$$diff(node) = \begin{cases} null & \text{if node == null} \\ ka_k x^{k-1} + diff(node.next) & \end{cases}$$

LinkedPolynomial : diff (เร็วและ iterative)

```
public class Polynomial {
  ...
  public Polynomial diff() {
    Polynomial resultPoly = new Polynomial();
    LinkedNode result = resultPoly.header;
    LinkedNode node = header.next;
    while( node != null ) {
      result.next = new LinkedNode(
                        node.coef * node.exp,
                        node.exp - 1,
                        null);
      result = result.next;
      node = node.next;
    }
    return resultPoly;
  }
  ...
```

มี n พจน์ ใช้เวลา O(?)

LinkedPolynomial : add




```
public class Polynomial {
  ...
  public Polynomial add(Polynomial that) {
    LinkedNode n1 = this.header.next;
    LinkedNode n2 = that.header.next;
    Polynomial r = new Polynomial();
    while( n1 != null && n2 != null ) {
      if (n1.exp > n2.exp) {
        r.addTerm(n1.coef, n1.exp); n1 = n1.next;
      } else if (n1.exp < n2.exp) {
        r.addTerm(n2.coef, n2.exp); n2 = n2.next;
      } else {
        r.addTerm(n1.coef + n2.coef, n1.exp);
        n1 = n1.next; n2 = n2.next;
      }
    }
    while( n1 != null ) {
      r.addTerm(n1.coef, n1.exp); n1 = n1.next;
    }
    while( n2 != null ) {
      r.addTerm(n2.coef, n2.exp); n2 = n2.next;
    }
    return r;
  }
```

มี n พจน์ ใช้เวลา O(?)

การบ้าน

- วิเคราะห์การทำงานของ add
- add ที่เขียน ทำงานช้า ปรับปรุงให้เร็วขึ้น
- เขียน multiply และ eval ให้ครบ
- สร้าง Polynomial โดยใช้ array

```
public static void main(String[] a) {
  Polynomial p = new Polynomial();
  p.setTerm(9, 10);
  p.setTerm(3, 2);
  p.setTerm(2, 1);
  p.setTerm(1, 0);
  System.out.println(p);
  Polynomial pp = p.add(p);

  System.out.println(pp);
  System.out.println(pp.diff());
  System.out.println(pp.multiply(p));
}
```

เขียน toString() ด้วย