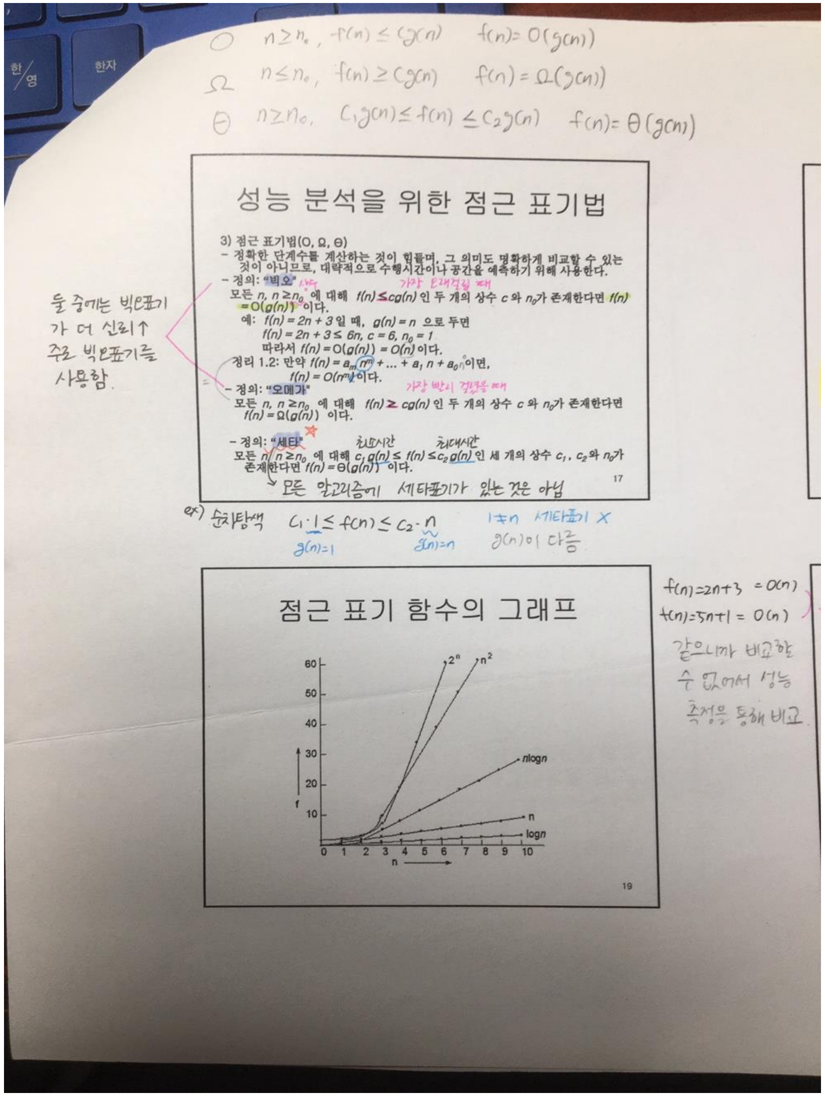
자료구조를 구조하자

2019.10.04

32153180 이상민

32162436 신창우 32163006 이건욱
32164420 조정민 32164959 허전진

학습 내용




```
Polynomial Polynomial::Multiple(Polynomial b)
	//a(x)(*this의 값)와 b(x)를 곱한 결과를 반환한다.
{
	Polynomial answer;//결과 값 저장

	//A항의 첫번째 항부터 B의 식과 차례대로 곱함
	for(int i=0;i<terms;i++)//A항의 개수만큼 반복문 실행
	{
		Polynomial c;//c의 식을 초기화

		for(int j=0;j<b.terms;j++)//B항의 개수만큼 반복문 실행
		{

c.NewTerm(termArray[i].coef*b.termArray[j].coef,termArray[i].exp+b.termArray
[j].exp);
		}//계수끼리는 곱하고 지수끼리는 더한 식을 c에 저장

		answer=answer.Add(c);//c의 식끼리 Add함수를 이용해서 더함
	}
	return answer;//결과 반환
}
```

신창우

학습 내용

〈 소스코드 〉

```
Polynomial Polynomial::multiply(Polynomial b) {
    Polynomial c;                                                    // 결과를 저장할 객체 생성
    int aPos, bPos;                                                  // 객체 A, B의 인덱스를 지정할 변수 선언

    for (aPos = 0; aPos < terms; aPos++) {                           // 다항식 A의 항의 수만큼 반복
        Polynomial load;                                             // 곱셈을 저장할 임시 객체 생성
        for (bPos = 0; bPos < b.terms; bPos++) {                     // 다항식의 곱셈 >> A항마다 B의 항의 수만큼 곱셈 반복
            float gop = termArray[aPos].coef * b.termArray[bPos].coef;   // 계수끼리 곱해준다
            int dut = termArray[aPos].exp + b.termArray[bPos].exp;       // 지수끼리 더해준다
            load.NewTerm(gop, dut);                                  // 계산된 계수와 지수를 임시 객체에 저장한다
        }
        c = c.Add(load);                                             // c에 값들을 누적한다
                                                                     // (지수가 같은 항들이 계속 나오므로 반복문안에 놓고 누적한다)
    }

    return c;
}
```

〈 실행화면 〉

```
다항식 A의 항의 수 : 2
다항식 A의 1 번째 항의 계수와 지수 : 3 2
다항식 A의 2 번째 항의 계수와 지수 : 3 0
다항식 B의 항의 수 : 2
다항식 B의 1 번째 항의 계수와 지수 : 5 2
다항식 B의 2 번째 항의 계수와 지수 : 2 1
다항식 A :
3 x^2 + 3 x^0
다항식 B :
5 x^2 + 2 x^1
곱셈 결과 :
15 x^4 + 6 x^3 + 15 x^2 + 6 x^1
계속하려면 아무 키나 누르십시오 . . .
```

```
다항식 A의 항의 수 : 3
다항식 A의 1 번째 항의 계수와 지수 : 3 5
다항식 A의 2 번째 항의 계수와 지수 : 6 3
다항식 A의 3 번째 항의 계수와 지수 : 3 1
다항식 B의 항의 수 : 2
다항식 B의 1 번째 항의 계수와 지수 : 3 3
다항식 B의 2 번째 항의 계수와 지수 : 4 0
다항식 A :
3 x^5 + 6 x^3 + 3 x^1
다항식 B :
3 x^3 + 4 x^0
곱셈 결과 :
9 x^8 + 18 x^6 + 12 x^5 + 9 x^4 + 24 x^3 + 12 x^1
계속하려면 아무 키나 누르십시오 . . .
```

```
다항식 A의 항의 수 : 2
다항식 A의 1 번째 항의 계수와 지수 : 1 2
다항식 A의 2 번째 항의 계수와 지수 : -3 0
다항식 B의 항의 수 : 3
다항식 B의 1 번째 항의 계수와 지수 : -1 5
다항식 B의 2 번째 항의 계수와 지수 : -3 3
다항식 B의 3 번째 항의 계수와 지수 : 5 0
다항식 A :
1 x^2 + -3 x^0
다항식 B :
-1 x^5 + -3 x^3 + 5 x^0
곱셈 결과 :
-1 x^7 + 9 x^3 + 5 x^2 + -15 x^0
계속하려면 아무 키나 누르십시오 . . .
```

```
다항식 A의 항의 수 : 3
다항식 A의 1 번째 항의 계수와 지수 : -1 3
다항식 A의 2 번째 항의 계수와 지수 : -1 2
다항식 A의 3 번째 항의 계수와 지수 : -1 1
다항식 B의 항의 수 : 3
다항식 B의 1 번째 항의 계수와 지수 : 1 3
다항식 B의 2 번째 항의 계수와 지수 : 1 2
다항식 B의 3 번째 항의 계수와 지수 : 1 1
다항식 A :
-1 x^3 + -1 x^2 + -1 x^1
다항식 B :
1 x^3 + 1 x^2 + 1 x^1
곱셈 결과 :
-1 x^6 + -2 x^5 + -3 x^4 + -2 x^3 + -1 x^2
계속하려면 아무 키나 누르십시오 . . .
```

```
다항식 A의 항의 수 : 5
다항식 A의 1 번째 항의 계수와 지수 : 5 5
다항식 A의 2 번째 항의 계수와 지수 : 4 4
다항식 A의 3 번째 항의 계수와 지수 : 3 3
다항식 A의 4 번째 항의 계수와 지수 : 2 2
다항식 A의 5 번째 항의 계수와 지수 : 1 1
다항식 B의 항의 수 : 1
다항식 B의 1 번째 항의 계수와 지수 : 1 0
다항식 A :
5 x^5 + 4 x^4 + 3 x^3 + 2 x^2 + 1 x^1
다항식 B :
1 x^0
곱셈 결과 :
5 x^5 + 4 x^4 + 3 x^3 + 2 x^2 + 1 x^1
계속하려면 아무 키나 누르십시오 . . .
```

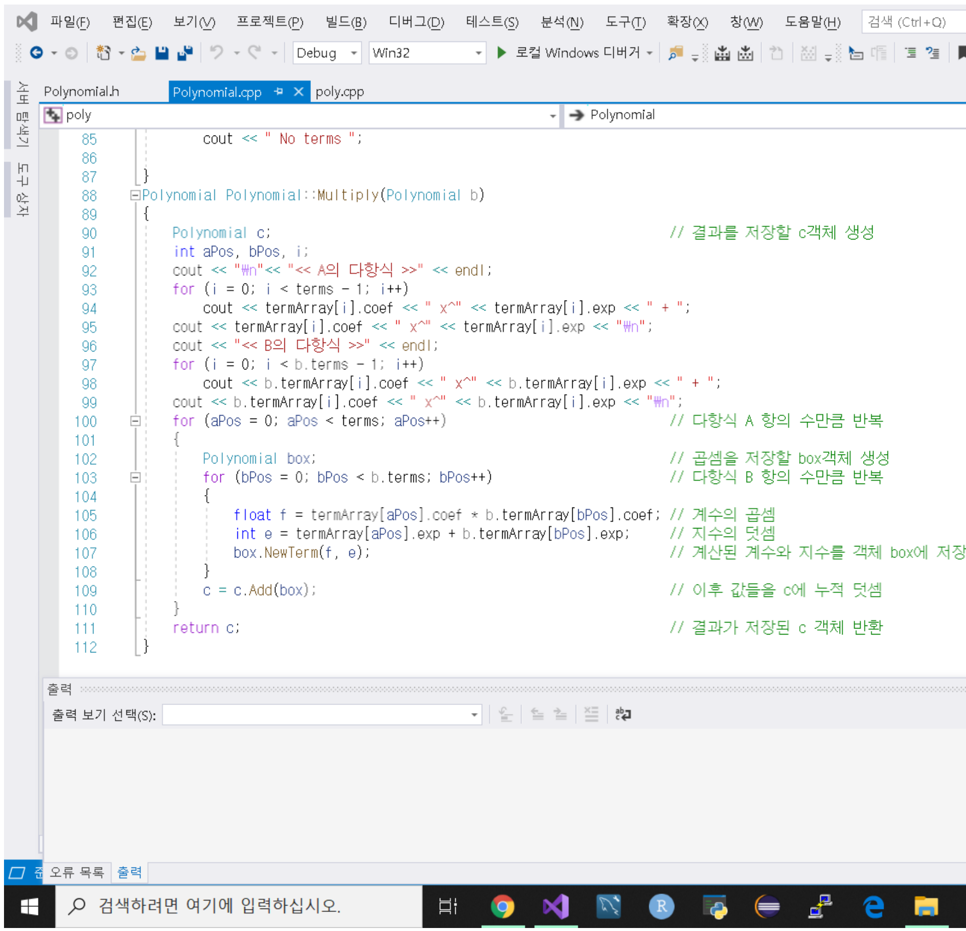
```
다항식 A의 항의 수 : 5
다항식 A의 1 번째 항의 계수와 지수 : 5 5
다항식 A의 2 번째 항의 계수와 지수 : 4 4
다항식 A의 3 번째 항의 계수와 지수 : 3 3
다항식 A의 4 번째 항의 계수와 지수 : 2 2
다항식 A의 5 번째 항의 계수와 지수 : 1 1
다항식 B의 항의 수 : 1
다항식 B의 1 번째 항의 계수와 지수 : 0 0
다항식 A :
5 x^5 + 4 x^4 + 3 x^3 + 2 x^2 + 1 x^1
다항식 B :
0 x^0
곱셈 결과 :
0 x^5 + 0 x^4 + 0 x^3 + 0 x^2 + 0 x^1
계속하려면 아무 키나 누르십시오 . . .
```

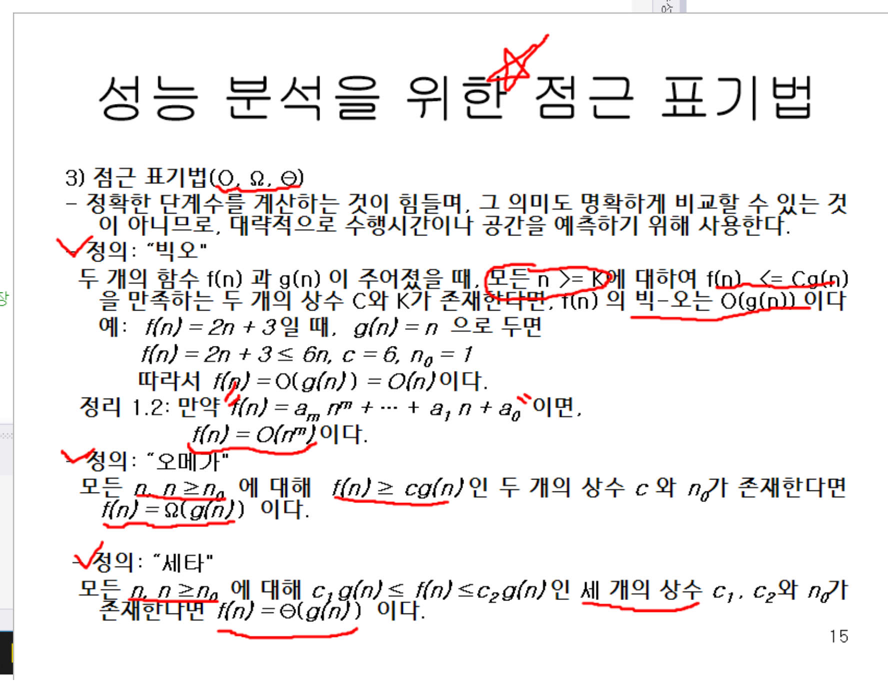
빅오 표기법은 알고리즘의 효율성을 표기해주는 표기법이다.
알고리즘의 효율성은 데이터 개수(n)가 주어졌을 때 덧셈,뺄셈,곱셈 같은 기본 연산의 횟수를 의미한다.
빅오 표기법은 보통 알고리즘의 시간 복잡도와 공간 복잡도를 나타내는데 주로 사용 된다.
(시간 복잡도는 알고리즘의 시간 효율성을 의미하고, 공간 복잡도는 알고리즘의 공간(메모리) 효율성을 의미한다.)
그런데 시간과 공간 복잡도를 나타내는 방법으로는 점근 표기법이라고 해서 빅오(Big−O), 빅오메가(big−Ω),빅세타(big−Θ) 표기법이 있다.
✓ 주로 사용하는 것은 빅오표기법이다.

$f(n) = 2n + 3 = O(n)$
$f(n) = 5n + 1 = O(n)$
−> 같기 때문에 비교할 수 없어 성능 측정을 통해 비교한다.
✓ 순차탐색 $c_1 \cdot 1 \le f(n) \le c_2 \cdot n \ (n \ne 1)$

이건욱

학습 내용





성능 분석을 위한 점근 표기법

3) 점근 표기법(O, Ω, Θ)

– 정확한 단계수를 계산하는 것이 힘들며, 그 의미도 명확하게 비교할 수 있는 것이 아니므로, 대략적으로 수행시간이나 공간을 예측하기 위해 사용한다.

정의: "빅오"

두 개의 함수 f(n) 과 g(n) 이 주어졌을 때, 모든 n >= K에 대하여 f(n) <= Cg(n)을 만족하는 두 개의 상수 C와 K가 존재한다면, f(n) 의 빅-오는 O(g(n)) 이다

예: $f(n) = 2n + 3$일 때, $g(n) = n$ 으로 두면

$f(n) = 2n + 3 \le 6n,\ c = 6,\ n_0 = 1$

따라서 $f(n) = O(g(n)) = O(n)$이다.

정리 1.2: 만약 $f(n) = a_m n^m + \cdots + a_1 n + a_0$이면, $f(n) = O(n^m)$이다.

정의: "오메가"

모든 $n,\ n \ge n_0$ 에 대해 $f(n) \ge cg(n)$인 두 개의 상수 c 와 n_0가 존재한다면 $f(n) = \Omega(g(n))$ 이다.

정의: "세타"

모든 $n,\ n \ge n_0$ 에 대해 $c_1 g(n) \le f(n) \le c_2 g(n)$인 세 개의 상수 c_1, c_2와 n_0가 존재한다면 $f(n) = \Theta(g(n))$ 이다.

15

조정민

학습 내용

```
    s += a[i];
    count++; // 지정문에 대한
  }
  count++;  // for문의 마지막 실행에 대한
  count++;  // return에 대한
  return s;
}
```

13

$f(n) = 2n+3$

$g(n) = n$

$c = 6,\ n_0 = 1$

$f(n) = 2n+3 \le 6n,\ n \ge 1$

$= O(g(n))$

$= O(n)$

성능 분석을 위한 점근 표기법

3) 점근 표기법(O, Ω, Θ)

– 정확한 단계수를 계산하는 것이 힘들며, 그 의미도 명확하게 비교할 수 있는 것이 아니므로, 대략적으로 수행시간이나 공간을 예측하기 위해 사용한다.

– 정의: "빅오" 최대시간 (worst case)
모든 n, $n \ge n_0$ 에 대해 $f(n) \le cg(n)$ 인 두 개의 상수 c 와 n_0가 존재한다면 $f(n) = O(g(n))$ 이다.
예: $f(n) = 2n + 3$ 일 때, $g(n) = n$ 으로 두면
$f(n) = 2n + 3 \le 6n$, $c = 6$, $n_0 = 1$
따라서 $f(n) = O(g(n)) = O(n)$ 이다.

$f(n) = 3n^3 + 2n^2 + n + 5$
$\rightarrow O(n^3)$ (빅오 표기)

⇒ 정리 1.2: 만약 $f(n) = a_m n^m + \dots + a_1 n + a_0$ 이면, $f(n) = O(n^m)$ 이다.

– 정의: "오메가" 최소시간 (best case)
모든 n, $n \ge n_0$ 에 대해 $f(n) \ge cg(n)$ 인 두 개의 상수 c 와 n_0가 존재한다면 $f(n) = \Omega(g(n))$ 이다.

– 정의: "세타"
모든 n, $n \ge n_0$ 에 대해 $c_1 g(n) \le f(n) \le c_2 g(n)$ 인 세 개의 상수 c_1, c_2와 n_0가 존재한다면 $f(n) = \Theta(g(n))$ 이다.

15

이진탐색

$C_1 \cdot 1 \le f$

소스 코드

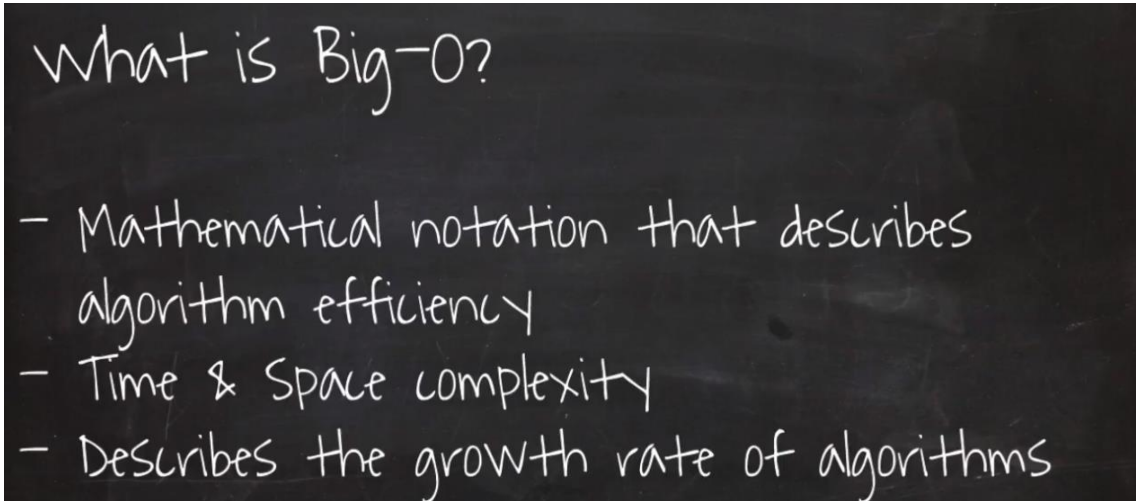
```
Polynomial Polynomial::Multiply(Polynomial b)
{
    Polynomial c;                                       //c = 반환할 결과 값
    int aPos, bPos;                                     //각 항의 수 선언
    for (aPos = 0; aPos < terms; aPos++) {              //A항의 수만큼 반복
        Polynomial res;                                 //res = 곱셈 결과 값
        for (bPos = 0; bPos < b.terms; bPos++) {        //B항의 수만큼 반복
            float f = termArray[aPos].coef * b.termArray[bPos].coef;//계수 곱셈
            int i = termArray[aPos].exp + b.termArray[bPos].exp;    //지수 덧셈
            res.NewTerm(f, i);                          //계수, 지수 res에 저장
        }
        c = c.Add(res);                                 //c에 결과값을 누적으로 덧셈
    }
    return c;                                           //결과 값 c 반환
}
```

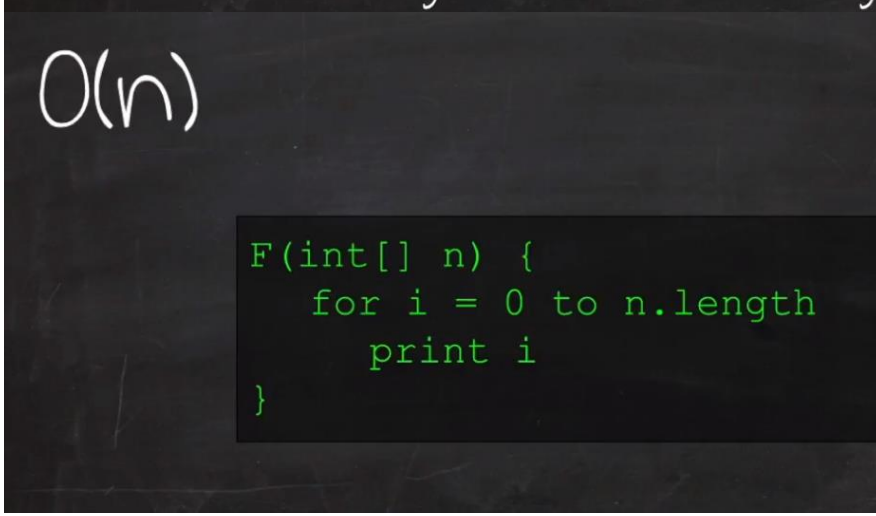
실행 화면

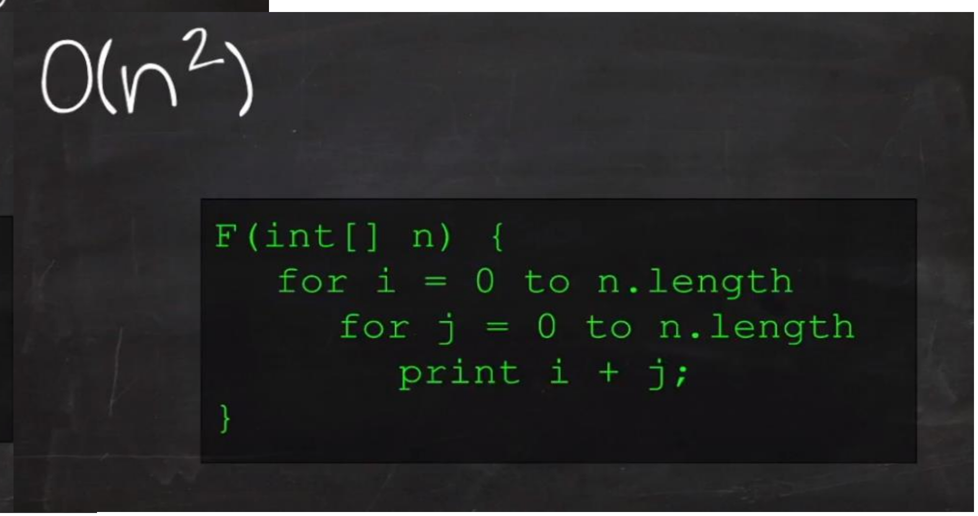
```
C:\Windows\system32\cmd.exe
다항식 A의 항의 수 : 2
다항식 A의 1 번째 항의 계수와 지수 : 4 1
다항식 A의 2 번째 항의 계수와 지수 : 2 3
다항식 B의 항의 수 : 2
다항식 B의 1 번째 항의 계수와 지수 : 1 5
다항식 B의 2 번째 항의 계수와 지수 : 2 3
* 다항식 A >> 4 x^1 + 2 x^3
* 다항식 B >> 1 x^5 + 2 x^3
* 다항식 A X 다항식 B >> 2 x^8 + 8 x^6 + 8 x^4
계속하려면 아무 키나 누르십시오 . . .
```

허전진

참고자료







https://www.youtube.com/watch?v=6Iq5iMCVsXA