배열과 구조

- * 자료구조의 구분
- 선형(linear) 과 비선형(non-linear)으로 구분한다.
- 자료구조내의 요소들이 순차적인 형식이면 선형, 아니면 비선형.
- 선형자료구조에는 배열과 연결 리스트가 있다.
- 배열: 순차적 메모리 주소에 의하여 요소들간의 관계를 표현하는 구조
- 연결리스트: 포인터를 사용하여 요소들간의 관계를 표현
- 비선형 자료구조: 트리(tree) 와 그래프(graph)

1. 배열의 특징

- 연속적 기억장소의 집합
- 대부분의 언어에서 제공하는 가장 단순한 구조적 자료형
- 동일한 자료형 (Same data type for elements)
- 선언시 크기지정. 크기보다 많은양의 자료 저장 ⇒ overflow
- 정적 자료형 (compile 시 크기를 알아야 하고, 실행 되는 동안 크기가 변하지 않는다)
- Set of mappings between index and values; <index, value>

장점: 이해 쉽고, 사용하기 편함, 자료저장이 용이(예: A[4]=10)

단점: 동일한 자료만 저장, 미리 크기 선언(필요이상 크기 선언시, 공간낭비, 많은 자료이동으로 삽입 삭제 느림.

• 배열의 연산

- i.length n
- ii. reading $(R \Rightarrow L, L \Rightarrow R)$
- iii. retrieve ith element, 0 <= i<n
- iv. update ith element's value, 0<= i<n
- v. insertion (*i번제 위치. 0<= I<= n*)
- vi. deletion (i번째 항목, 0 i<n)

2. 기호 다항식의 조작

 $A(x) = a_n x^n + a_{n-1} x^{n-1} + ... + a_1 x^1 + a_0 x^0$

 ax^e $a: coefficient <math>a_n >= 0$

e: exponent – unique

x: variable x

- 차수(degree): 다항식에서 가장 큰 지수

* 다항식의 표현 (1): 지수들을 내림차순으로 정돈

$$A = (n, a_n, a_{n-1}, \ldots, a_1, a_0)$$

degree of A n+1 coefficients

예) $3x^4 + 5x^2 + 6x + 4$ 의 경우 A = (4, 3, 0, 5, 6, 4)

문제점)
$$A(x) = x^{1000} + 1$$
 : n = 1000
A = (1000, 1, $0, ..., 0$, 1) : 1002 elements

* 다항식의 표현 (II)

$$A(x) = b_{m-1}x^{em-1} + b_{m-2}x^{em-2} + \dots + b_{o}x^{e0}$$
 $Where \ b_i \neq 0 \ , 0 \le i \le m-1, \ e_{m-1} > e_{m-2} > \dots > e_o \ge 0$
 $A = (m, e_{m-1}, b_{m-1}, e_{m-2}, b_{m-2}, \dots, e_0, b_0)$
 $no. \ of \ non-zero \ terms$
 $A(x) = x^4 + 10x^3 + 3x^2 + 1 \qquad A = (4, 4, 1, 3, 10, 2, 3, 0, 1)$
 $A(x) = x^{1000} + 1 \qquad A = (2, 1000, 1, 0, 1)$

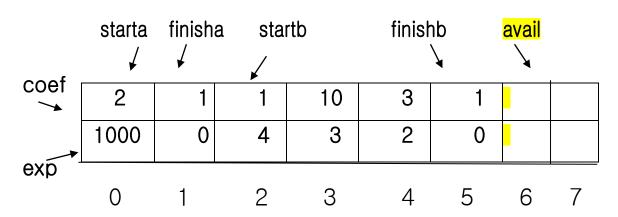
MAX_TERMS 100 /* 항 배열의 크기 */

coef	3	5	6	4
expon	4	2	1	0

ex) 두개의 다항식 A(x), B(x)

$$A(x) = 2x^{1000} + 1$$

B(x) = $x^4 + 10x^3 + 3x^2 + 1$



startA = 0, finishA = 1, startB = 2, finishB = 5, Avail = 6

다항식 덧셈 D = A + B

```
void padd (int starta, int finisha, int startb, int finishb, int *startd,
int *finishd);
{ /* A(x) 와 B(x)를 더하여 D(x)를 생성한다 */
  float coefficient:
                       *startd = avail:
while (starta <= finisha && startb <= finishb)
   switch(COMPARE(terms[starta].expon, terms[startb].expon))
     case -1: /* a의 expon이 b의 expon보다 작은 경우 */
           attach(terms[startb].coef, terms[startb].expon);
           startb++; break;
     case 0: /* 지수가 같은 경우 */
           coefficient= terms[starta].coef+terms[startb].coef;
           if(coefficient)
               attach(coefficient, terms[starta].expon);
               starta++; startb++; break;
    case 1: /* a의 expon이 b의 expon보다 큰 경우 */
          attach(terms[starta].coef, terms[starta].expon);
          starta++;
   }
 /* A(x)의 나머지 항들을 첨가하다 */
  for(; starta <= finisha; starta++)</pre>
       attach(terms[starta].coef, terms[starta].expon);
 /* B(x)의 나머지 항들을 첨가한다 */
 for(; startb <= finishb; startb++)</pre>
       attach(terms[startb].coef,terms[startb].expon);
  *finishd = avail-1;
 => Avail? 测点 ~? > + 划以品(+1 刈)
```

```
void attach(float coefficient, int exponent)
{ /* 새 항을 다항식에 첨가한다. */
    if (avail >= MAX_TERMS) {
        fprintf(stderr, "다항식에 항이 너무 많다.");
        exit(1);
    }
    terms[avail].coef = coefficient;
    terms[avail++].expon = exponent;
}
```

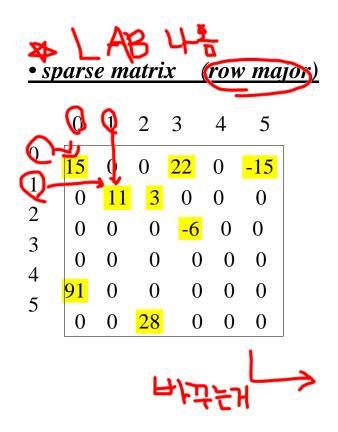
3. 희소행렬 (Sparse Matrix)

1) 개요

```
m \times n \ matrix \ A \equiv A[MAX\_ROWS][MAX\_COLS]
```

number of non-zero elements / total elements << small

- 효율적 기억장소 사상
 - i) <i, j, value>: 3-tuples (triples)
 - ii) no. of rows
 - iii) no. of columns
 - iv) no. of non-zero elements
 - v) ordering (column major or row major)



(row) (col) (value)

6	6	8
0	0	15
0	3	22
0	5	-15
1	1	11
1	2	3
2	3	-6
4	0	91
5	2	28

Sparse Matrix 'a'



for문 두개 써서 출력하면됨



Sparse Matrix 'b'

희소 행렬의 전치 (Transpose)

```
void transpose( SMarray a[], SMarray b[])
/* a 를 전치시켜 b 를 생성, 예:(0,3,22) -> (3,0,22) */
  int i, j, currentb;
   b[0].row = a[0].col;
   b[0].col = a[0].row;
   b[0].value = a[0].value;
  if (a[0].value > 0) { /* 0 이 아닌 행렬 */
    currentb = 1;
    for (i =0; i <a[0].col; i++) /* a 에서 열별로 전치*/
      for (j = 1; j \le a[0].value; j++)
           /* 현재의 열로부터 원소를 찾는다. */
           if (a[j].col == i) {
             /*현재 열에 있는 원소를 b에 첨가 */
             b[currentb].row = a[j].col;
             b[currentb].col = a[j].row;
             b[currentb].value=a[i].value;
             currentb++;
         }
 /* for (j=0; j<col; j++)
                                */
     for( i=0; i<row; i++)
                                */
                                */
 /*
        b [j][i] = a[i][j];
```