/ /

(Section - 6)

(Sparse Matrix)

× **Sparse Matrix**

Representation :- ① Coordinate List / 3-column Rep.

② compressed sparse row

A Spare matrix is a matrix where most of the elements are zero

Ex :-
$$\begin{array}{c|cccc} & 0 & 1 & 2 & 3 \\ \hline 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 2 & 0 & 5 & 0 & 2 \\ 3 & 9 & 0 & 0 & 6 \\ 4 & 7 & 0 & 0 & 0 \end{array}$$

4X3

Coordinate list / 3- Column Rep.

Total no of Rows ← 4

3 → Total no of columns

6 → no of elements

Row	Column	Value
4	3	6
0	1	1
2	1	5
2	3	2
3	0	9
3	3	6
4	0	7

Compressed Sparse row (CSR)

$$\begin{array}{c|cccc} & 0 & 1 & 2 & 3 \\ \hline 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 2 & 0 & 5 & 0 & 2 \\ 3 & 9 & 0 & 0 & 6 \\ 4 & 7 & 0 & 0 & 0 \end{array}$$

4x3

stores all non-zero element

A [1, 5, 2, 9, 6, 7]

JA [1, 1, 3, 0, 3, 0]

↳ stores the column index

IA [0, 1, 1, 3, 5, 6]

* Addition of Sparse Matrix :-

One method is simple just add the elements (zero & non-zero's)

using Coordinate List

$$A = \begin{array}{c|cccccc} & 1 & 2 & 3 & 4 & 5 & 6 \\ \hline 1 & 0 & 0 & 0 & 6 & 0 & 0 \\ 2 & 0 & 7 & 0 & 0 & 0 & 0 \\ 3 & 0 & 2 & 0 & 5 & 0 & 0 \\ 4 & 0 & 0 & 0 & 0 & 0 & 0 \\ 5 & 4 & 0 & 0 & 0 & 0 & 0 \end{array}$$

$$B = \begin{array}{c|cccccc} & 1 & 2 & 3 & 4 & 5 & 6 \\ \hline 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 2 & 0 & 3 & 0 & 0 & 5 & 0 \\ 3 & 0 & 0 & 2 & 0 & 0 & 7 \\ 4 & 0 & 0 & 0 & 9 & 0 & 0 \\ 5 & 8 & 0 & 0 & 0 & 0 & 0 \end{array}$$

A

0	1	2	3	4	5
5	1	2	3	3	5
6	4	2	2	4	1
5	6	7	2	5	4

B

0	1	2	3	4	5	6
5	2	2	3	3	4	5
6	2	5	3	6	4	1
6	3	5	2	7	9	8

C

0	1	2	3	4	5	6	7	8	9
5	1	2	2	3	3	3	3	4	5
6	4	2	5	2	3	4	6	4	1
9	6	10	5	2	2	5	7	9	12

* Array Representation

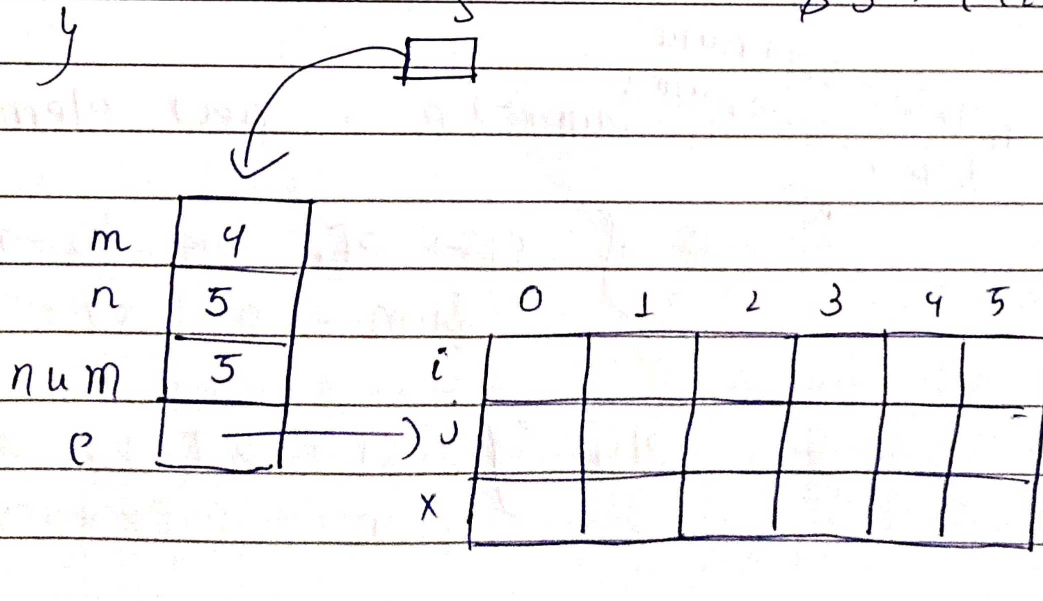
$$A = \begin{bmatrix} 0 & 0 & 7 & 0 & 0 \\ 2 & 0 & 0 & 5 & 0 \\ 9 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 4 \end{bmatrix}$$

4x5

	0	1	2	3	4	5
i	4	1	2	2	3	4
j	5	3	1	4	1	5
x	5	7	2	5	9	4

Code :-

```
struct element {
    int i;
    int j;
    int x;
}
struct sparse {
    int m;
    int n;
int num;
struct element *e;
}

void create (struct sparse *s)
{
   int i;
   printf ("Enter Dimensions:");
   scanf ("%d %d", &s->m, &s->n);

  printf ("No of non-zero elements:");
  scanf ("%d", &s->num);

 s->e = new element[s->num];

 printf ("Enter all elements:");

 for (i=0; i< s->num; i++)
   scanf ("%d %d %d", &s->e[i].i
                       &s->e[i].j
                       &s->e[i].x);
}
```



* **Program for adding Sparse Matrix :**

$$\begin{bmatrix} 0 & 0 & 3 & 0 & 0 \\ 4 & 0 & 0 & 0 & 7 \\ 0 & 0 & 5 & 0 & 8 \\ 0 & 6 & 0 & 0 & 0 \end{bmatrix} \qquad \begin{bmatrix} 0 & 0 & 0 & 0 & 2 \\ 0 & 5 & 0 & 0 & 6 \\ 4 & 0 & 8 & 0 & 0 \\ 0 & 0 & 0 & 0 & 9 \end{bmatrix}$$

S1

m	4
n	5
num	6
e	→

1	2	2	3	3	4
3	1	5	3	5	2
3	4	7	5	8	6

S2

m	4
n	5
num	6
e	→

1	2	2	3	3	4
5	2	5	1	3	5
2	5	6	4	8	9

```
add (Sparse *s1, Sparse *s2)
{
    sparse * sum;
    if (s1->m != s2->m || s1->n != s2->n)
        return 0;

    sum = new sparse;
    sum->m = s1->m; sum->n = s1->n;
    sum->e = new element [s1->num +
                          s2->num];
while (i<s1->num && j<s2->num)
    if (s1->e[i].i < s2->e[j].i)
        sum->e[k++] = s1->e[i++];

    else if (s1->e[i].i > s2->e[j].i)
        sum->e[k++] = s2->e[j++];
```

```
else
{
    if (s1->e[i].j < s2->e[j].j ) sum->e[k++]
                                   =s1->e[i++];

else if (s1->e[i].j > s2->e[j].j ] sum->e[k++]
                                   =s2->e[j++];

else { sum->e[k] = s1->e[i++];
       sum->e[k++].x+= s2->e[j++].x
     }
}
```

Polynomial Representation :-

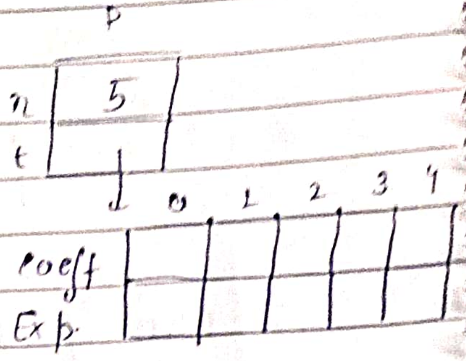
$$p(x) = 3\overset{\text{coeff}}{}x^{\overset{\text{exponent}}{\boxed{5}}} + 2x^4 + 5x^2 + 2x + 7$$

n= 5

Coeff	3	2	5	2	7
Exp.	5	4	2	1	0

```
struct Term {                     struct Poly {

    int coeff;                        int n;
    int exp;                          struct Term *t;
}                                 }
```

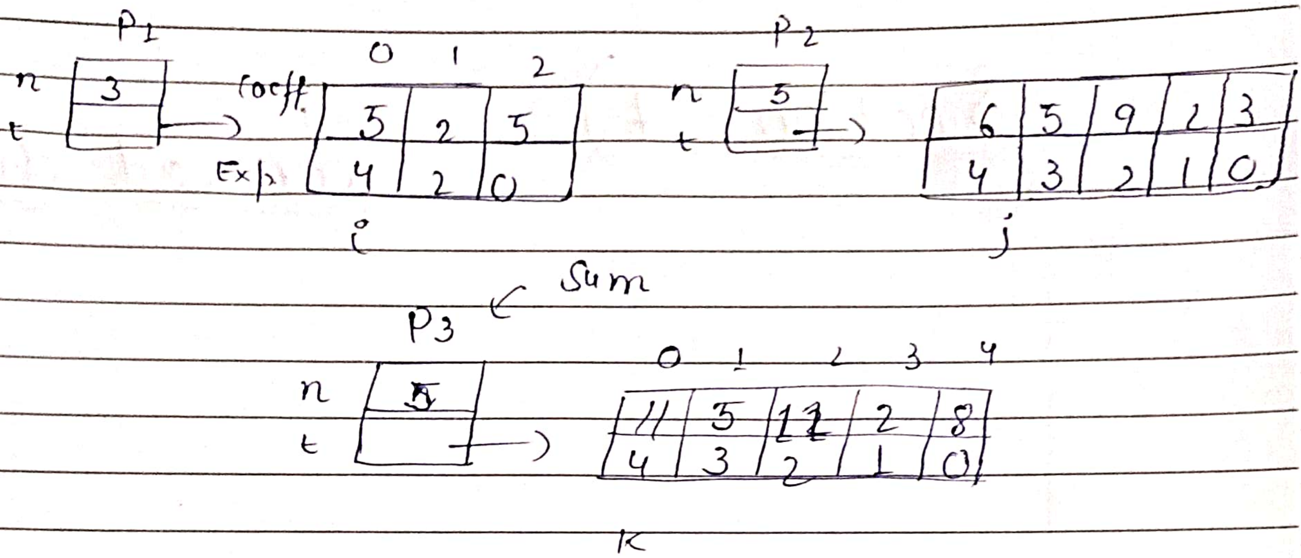
```
Struct Poly P;

printf ("Enter no of non-zero
                          terms");
Scanf ("%d", & p.n);

p.t = new Term [p.n];

printf("Enter polynomial terms");
for (i=0; i<p.n; i++)
  {
     printf ("Term no %d", i+1);
     Scanf ("%d %d", & p.t[i].coeff,
                          & p.t [i].exp);
  }

for (i=0; i<p.n; i++)
  {
     sum += p.t[i].coeff * pow (x, p.t[i].Exp);
  }

return sum;
```



* Polynomial Representation

$p1(x) = 5x^4 + 2x^2 + 5$

$p2(x) = 6x^4 + 5x^3 + 9x^2 + 2x + 3$



coeff	5	2	5
Exp	4	2	0

6	5	9	2	3
4	3	2	1	0

11	5	11	2	8
4	3	2	1	0

```
i=0, j=0, k=0;

while (i<p1.n && j<p2.n)
{
  if (p1.t[i].Exp > p2.t[j].Exp)

       p3.t [k++] = p1.t[i++];

  else if (p2.t [j] Exp > p1.t [i].Exp)

       p3.t [k++] = p2.t [j++];

  else

       p3.t [k].Exp = p1.t [i].Exp;
       p3.t [k++].coeff = p1.t [i++].coeff
                          + p2.t [j++].
                                      coeff.
}
```