چند جمله‌ای ها

تعریف. یک چند جمله ای مجموعه ای از جملات است که هر جمله به صورت ax^e می باشند . به طوریکه a ضریب ، e توان و x یک متغیر است.

مثال : $A(x) = 3x^{20} + 2x^5 + 4$

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

بیشترین توان یک چند جمله ای **درجه** آن نام دارد . ضرایبی که صفر می باشند نشان داده نمی شوند.

عملیاتی که روی چند جمله ای ها می توان انجام داد :

1-جمع دو چند جمله ای

2- ضرب دو چند جمله ای

3-تقسیم یک چند جمله ای بر یک چند جمله‌ای

4-تفریق دو چند جمله ای

5- حذف یک جمله از چند جمله ای

6- اضافه کردن یک جمله به چند جمله ای و ....

برای نمایش چند جمله ای ها می توان از ساختمان و آرایه کمک گرفت . البته قبل از ذخیره کردن اطلاعات توانها را صورت نزولی مرتب می کنیم .

روش 1 : ذخیره کردن ضرایب و درجه چند جمله ای .

```
#define  MAX _TERMS  101
// MAX_TERMS حداکثر تعداد جملات در یک چند جمله ای را نشان می‌دهد
typedef struct {
      int   degress ,
      float coef[MAX_TERMS];
} poly;
```

با این ساختار ما یک چند جمله ای از درجه 100 را می توانیم ذخیره کنیم .

مزایا :

ما توانها را ذخیره نمی کنیم ، الگوریتم جمع آن ساده است . این روش برای حالتی خوب است که تعداد ضرایب صفر یا کم باشد . (برای ذخیره چند جمله ای با توان بالا در این روش محدود هستیم) .

مثال : $A(x) = 3x^3 + 2x + 1$

0	1	2	3	...	100
1	2	0	3	0	0

روش دوم :ذخیره کردن توان و ضریب .

```
typedef       struct {
      float    coef ;
      int    exp ;
```

```
} term;
term  terms[MAX_TERMS];
// آرایه ای برای ذخیره جملات چند جمله‌ای
typedef struct{
      int start;  // اندیس شروع چند جمله ای در آرایه
      int end; // اندیس پایان چند جمله ای در آرایه
} poly;
```

مزیت : یک چند جمله ای که ضرایب صفر آن زیاد باشد می توانیم ذخیره کنیم .

$$A(x) = 2x^{1000} + 1$$
$$B(x) = x^4 + 10x^3 + 3x^2 + 1$$

۲	۱	۱	۱۰	۳	۱			
۱۰۰۰	۰	۴	۳	۲	۰			

avail

avail نشان دهنده اولین محل خالی در آرایه جهت قرار گرفتن جمله جدید می باشد.

الگوریتم جمع دو چند جمله ای A, B :

تابع جمع دو چند جمله ای:

تابع جمع دو چند جمله ای نیاز به استفاده از یک تابع به نام attach دارد. ورودی تابع attach یک ضریب و یک توان است. تابع attach ضریب و توان را در آرایه عمومی چند جمله ای ها اضافه می کند.

```
void    attach ( float  c , int e ) {
      if( avail < MAX_TERMS) {
            terms[avail].coef=c;
            terms[avail].exp=e;
            avail ++;
            return ;
      }
      printf("Error : out  of memory");
}
void  padd ( poly a, poly b, poly *c )
{
      int i,j;
      (*c).start=avail;
      i=a.start;   j=b.start;
      while ( i<=a.end && j<=b.end)
      {
            if (terms[i].exp > terms [j].exp ) {
                  attach(terms [i].coef, terms [i].exp);
                  i++;
            } else
            if (terms [i].exp < terms [j].exp ) {
                  attach(terms [j].coef, terms [j].exp);
                  j++;
```

```
            } else     {
                  float temp= terms[i].coef + terms[j].coef;
                  if (temp != 0) attach(temp,terms[i].exp);
                  i++;
                  j++;
            }
      }
      for(;i<=a.end;i++)
             attach(terms[i].coef,terms[i].exp);
      for(;j<=b.end;j++)
            attach(terms[j].coef,terms[j].exp);
      (*c).end=avail - 1;
}
```

اگر m تعدد جملات غیر صفر چند جمله ای A(x) باشد و n تعداد جملات غیرصفر چند جمله ای B(x) باشد آنگاه مرتبه اجرایی این الگوریتم برابر است با O(m+n) .

ماتریس اسپارس :

(یا ماتریس تنک ، ماتریسی که صفرهای زیادی داشته باشد .)

برای ذخیره این ماتریس اسپارس از یک جدول که از سه ستون تشکیل شده است استفاده می کنیم .

اولین سطر در این جدول اطلاعات مربوط به تعداد سطرها ، تعداد ستونها و تعداد عناصر مخالف با صفر است . از سطر 1 به بعد اطلاعات ماتریس ذخیره می شود .

	ستون5		طریقه ذخیره سازی ماتریس اسپارس
سطر 6	$\begin{bmatrix} 0 & 15 & 0 & 19 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 12 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 18 & 0 & 0 & 0 & 0 \end{bmatrix}$	→	

Row	Col	value
6	5	4
0	1	15
0	3	19
2	2	12
5	0	18

```
#define    MAX_TERMS    101
typedef  struct  {
     int row;
     int col;
     float value;
}   term ;
term    sp[ MAX_TERMS ];
```

عملیاتی که روی ماتریس اسپارس می توان تعریف کرد می تواند شامل محاسبه :

1- ترانهاده ماتریس اسپارس .

2- جمع دو ماتریس اسپارس و تفریق دو ماتریس اسپارس .

3- ضرب دو ماتریس اسپارس .

مثال - یک ماتریس $A_{n\times n}$ در نظر بگیرید و الگوریتم ترانهاده آن را بنویسید .

```
void   transpose    ( int  a [n][n])
   {
       int  i , j , temp ,
       for ( i=0 , i<n , i++ )
             for (j=0 , j<i , j ++ )
                {
                     temp = a [i][j] ;
                     a [i][j] = a [j][i];
                     a [j][i] = temp;
                  }
 }
```

پیچیدگی این الگوریتم:

در تابع فوق تعداد اجرا شدن حلقه for داخلی بستگی به مقدار I در حلقه بیرونی دارد بنابر این تعداد اجرا شدن دستورات داخل حلقه for درونی برابر است با:

تعداد اجرا= $0 + 1 + 2 + 3 + . . . + n-1 = n(n-1) /2$

بسادگی می توان دریافت مرتبه اجرایی الگوریتم فوق $O(n^2)$ می باشد.

ماتریس اصلی A			ماتریس ترانهاده B		
Row	Col	value	Row	Col	value
6	5	4	5	6	4
0	1	15	0	5	18
0	3	19	1	0	15
2	2	12	2	2	12
5	0	18	3	0	19

الگوریتم ترانهاده ماتریس اسپارس :

```
void   transpose2  ( terms  a[] , terms b[])
{
      b[0].row = a[0].col ;
       b[0].col = a[0].row ;
       b[0].value = a[0].value ,
       int  n=a[0].value ;
       int  i,cb = 1 ;

       for ( i=0 ; j<a [0].col ; i++)
           {
               int j;
                for ( j=1 ; j <= n ; j++)
                    if ( a [j] . col == i )   {
                      b [cb].row = a[j].col;
                      b [cb].col = a[j].row;
                      b [cb].value = a[j].value;
                      cb ++ ;
                   }
           }
      }
```

مرتبه زمانی :

مرتبه زمانی الگوریتم فوق عبارتند از $n \times col$ که n تعداد عناصر غیر صفر ماتریس بوده و

col تعداد سطرهای ماتریس را نشان می دهد

در بدترین حالت : تعداد عناصر غیرصفر برابر تعداد عناصر آرایه a است، که برابر است با

row.col پس مرتبه زمانی الگوریتم فوق در بدترین حالت برابر $O(row.col^2)$ می باشد.

تمرین : الگوریتمی برای جمع دو ماتریس اسپارس بنویسید و مرتبه زمانی آن را بدست آورید .