به نام خدا

ساختمان داده ها

جلسه سوم

دانشگاه صنعتی همدان

گروه مهندسی کامپیوتر

نیم سال دوم 98-1397

آرایه ها

- ADT ارایه
- نمایش ارایه ها
- ADT چند جمله ایها
- ADTماتریسهای خلوت

استفاده از ارایه برای پیاده سازی ADTهای دیگر- چند جمله ایها

- معمولا از آرایه که خود یک ADT است برای پیاده سازی ADT های دیگر استفاده می شود. یکی از مثالهای خوب برای این کاربرد؛ چند جمله ایها هستند. می خواهیم ADT پیاده کنیم که در آن چند جمله ایها را ذخیره کرده و یک سری اعمال روی انها انجام دهیم.

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

Class Polynomial{

Objects: a set of ordered pairs of <e_i,a_i> where a_i in *Coefficients* and e_i in *Exponents*, e_i are integers >= 0

Methods:

for all *poly*, *poly1*, *poly2* ∈ *Polynomial*, *coef* ∈ *Coefficients*, *expon* ∈ *Exponents*

Polynomial Zero( ) ::= **return** the polynomial *p(0)*

Boolean IsZero(*poly*) ::= **if** (*poly*) **return** *FALSE*
else return *TRUE*

Coefficient Coef(*poly*, *expon*) ::= **if** (*expon* ∈ *poly*) **return** its coefficient **else return** Zero

Exponent Lead_Exp(*poly*) ::= **return** the largest exponent in ***poly***

Polynomial Attach(*poly*,*coef*, *expon*) ::= **if** (*expon* ∈ *poly*) **return** error
else return the polynomial poly with the term <*coef*, *expon*> inserted

Polynomial Remove(*poly*, *expon*) ::= **if** (*expon* ∈ *poly*) **return** the polynomial *poly* with the term whose exponent is *expon deleted*
else return error

Polynomial Add(*poly1*, *poly2*) ::= **return** the polynomial *poly*1 +*poly*2

Polynomial Mult(*poly1*, *poly2*) ::= **return** the polynomial *poly*1 • *poly*2

}

پیاده سازی ADT چند جمله ای ها

- نمایش چند جمله ایها

```
class term {
friend Polynomial;
private:
float  coef;
Int  exp;
}

class Polynomial{
private:
static term termArray[Max];
static int free;
int start, finish;
```

Store pairs of exponent and coefficient

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

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

advantage: less space

disadvantage: longer code

	starta	*finisha*	*startb*			*finishb*	*avail*
coef	2	1	1	10	3	1	
exp	1000	0	4	3	2	0	
	0	1	2	3	4	5	6

```
Polynomial Polynomial::Add(Polynomial B)
{
polynomial C; int a=Start, b=B.Start, C.Start=free; float c;
   While((a<=Finish)&&(b<=B.Finish))
       switch(compare(termArray[a].exp,termArray[b].exp))
           { case  ‘=‘:  c=termArray[a].coef+termArray[b].coef ;
                          if ( c ) NewTerm(c, termArray[a].exp);
                          a++; b++;
                           break;
             case ‘<‘ :
                         NewTerm(termArray[b].coef, termArray[b].exp);
                         b++;
                         break;
             cas ‘>’:
                          NewTerm(termArray[b].coef, termArray[a].exp);
                           a++
            }
    for(;a<=Finish;a++)    NewTerm(termArray[a].coef,termArray[a].exp);
    for(;b<=B.Finish;b++)    NewTerm(termArray[b].coef,termArray[b].exp);
    C.Finish=free-1;
Return C
}
```

```
void Polynomial::NewTerm(float c, int e)
{
  if (free>=MaxTerms)
      {
         cout<<“overflow”;
         return;
       }
 termArray[free].coef=c;
 termArray[free].exp=e;
 free++
}
```

- تحلیل زمانی جمع ؟
- معایب این روش ؟

ماتریسهای خلوت

- ماتریس یک شی ریاضی است که برای پیاده سازی آن از آرایه های دو بعدی استفاده می شود.

اما در عمل گاهی وقتها ماتریسهای داریم که بسیاری از عناصر انها صفر هستند که به آنها ماتریس خلوت می گوییم. بنابراین استفاده از ارایه دو بعدی برای ذخیره انها منطقی نمی باشد و فضای زیادی اشغال می کند. ما برای این نوع ماتریسها یک ADT تعریف و پیاده می کنیم.

$$
\begin{array}{c} \\ \text{row 1} \\ \text{row 2} \\ \text{row 3} \\ \text{row 4} \\ \text{row 5} \end{array}
\begin{array}{c}
\begin{array}{ccc} \text{col 1} & \text{col 2} & \text{col 3} \end{array} \\
\begin{bmatrix} -27 & 3 & 4 \\ 6 & 82 & -2 \\ 109 & -64 & 11 \\ 12 & 8 & 9 \\ 48 & 27 & 47 \end{bmatrix}
\end{array}
$$

5*3

15/15

$$
\begin{array}{c} \\ \text{row0} \\ \text{row1} \\ \text{row2} \\ \text{row3} \\ \text{row4} \\ \text{row5} \end{array}
\begin{array}{c}
\begin{array}{cccccc} \text{col1} & \text{col2} & \text{col3} & \text{col4} & \text{col5} & \text{col6} \end{array} \\
\begin{bmatrix} 15 & 0 & 0 & 22 & 0 & -15 \\ 0 & 11 & 3 & 0 & 0 & 0 \\ 0 & 0 & 0 & -6 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 91 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 28 & 0 & 0 & 0 \end{bmatrix}
\end{array}
$$

6*6

8/36

sparse matrix
data structure?

ADT ماتریس خلوت

Class spars{

Objects: a set of triples, <*row, column, value*>, where *row* and *column* are integers and form a unique combination, and *value* comes from the set *item.*

Methods:

for all *a, b* ∈ *Sparse_Matrix*, *x* ∈ *item, i, j, max_col, max_row* ∈ *index*

Sparse_Marix Create(*max_row, max_col*) ::=

return a *Sparse_matrix* that can hold up to *max_items* = *max _row* × *max_col* and whose maximum row size is *max_row* and whose maximum column size is *max_col.*

Sparse_Matrix Transpose(*a*) ::=

return the matrix produced by interchanging the row and column value of every triple.

Sparse_Matrix Add(*a, b*) ::=

if the dimensions of a and b are the same
return the matrix produced by adding corresponding items, namely those with identical *row* and *column* values.
else return error

Sparse_Matrix Multiply(*a, b*) ::=

if number of columns in a equals number of rows in **b**
return the matrix *d* produced by multiplying a by *b* according to the formula: $d[i][j] = \Sigma(a[i][k] \cdot b[k][j])$ where $d(i, j)$ is the (i,j)th element
else return error.

}

پیاده سازی ADT ماتریسهای خلوت

- نمایش

	row	col	value
[0]	0	0	15
[1]	0	3	22
[2]	0	5	-15
[3]	1	1	11
[4]	1	2	3
[5]	2	3	-6
[6]	4	0	91
[7]	5	2	28

$$\begin{bmatrix} 15 & 0 & 0 & 22 & 0 & -15 \\ 0 & 11 & 3 & 0 & 0 & 0 \\ 0 & 0 & 0 & -6 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 91 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 28 & 0 & 0 & 0 \end{bmatrix}$$

ماتریسهای خلوت - نمایش

```
class MatrixTerm{
friend class SparseMatrix
private:
int row,col,value;
}

class sparseMatrix{
private:
int Rows,Cols,Terms;
MatrixTerm  smArray[Max];
```