به نام خدا

ساختمان داده ها

جلسه چهارم

دانشگاه صنعتی همدان

گروه مهندسی کامپیوتر

نیم سال دوم 98-1397

ADTماتریسهای خلوت

- نمایش ماتریسهای خلوت
- اعمال ماتریسهای خلوت
  - ترانهاده
  - جمع
  - ضرب

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];
```

اعمال ماتریسهای خلوت - ترانهاده

	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

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

الگوریتم ترانهاده و پیچیدگی آن

```
SparseMatrix SparseMatrix::Transpose( )
{
   SparseMatrix   b;
   b.Rows = Cols;   b.Cols = Rows;   b.Terms = Terms;
  if (Terms > 0)
    {     int CurrentB = 0;
          for (int c = 0; c < Cols; c++)
             for (int i = 0; i <  Terms; i++)
                if (smArray[i].col == c)
                {
                   b.smArray[CurrentB].row = c;
                   b.smArray[CurrentB].col = smArray[i].row;
                   b.smArray[CurrentB].value = smArray[i].value;
                   CurrentB++;
                }
       }
return b;
}
```

تحلیل زمانی؟

ترانهاده سریع

```
SparseMatrix  SparseMatrix::FastTranspose()
{
   int *RowSize = new int[Cols];
   int *RowStart= new int[Cols];
   SparseMatrix   b;
   b.Rows = Cols; b.Cols = Rows; b.Terms = Terms;
   if (Terms > 0)
 {
    for (int i = 0; i < Cols; i++) RowSize[i] = 0;
    for (i = 0; i < Terms; i++) RowSize[smArray[i].col]++;

    RowStart[0] = 0;
    for (i = 1; i < Cols; i++) RowStart[i] = RowStart[i-1] + RowSize[i-1];
    for (i = 0 ; i < Terms; i++)
    {
       int j = RowStart[smArray[i].col];
       b.smArray[j].row = smArray[i].col;
       b.smArray[j].col = smArray[i].row;
       b.smArray[j].value = smArray[i].value;
       RowStart[smArray[i].col]++;
    }
 }
 return b;
}
```

ضرب ماتريسها

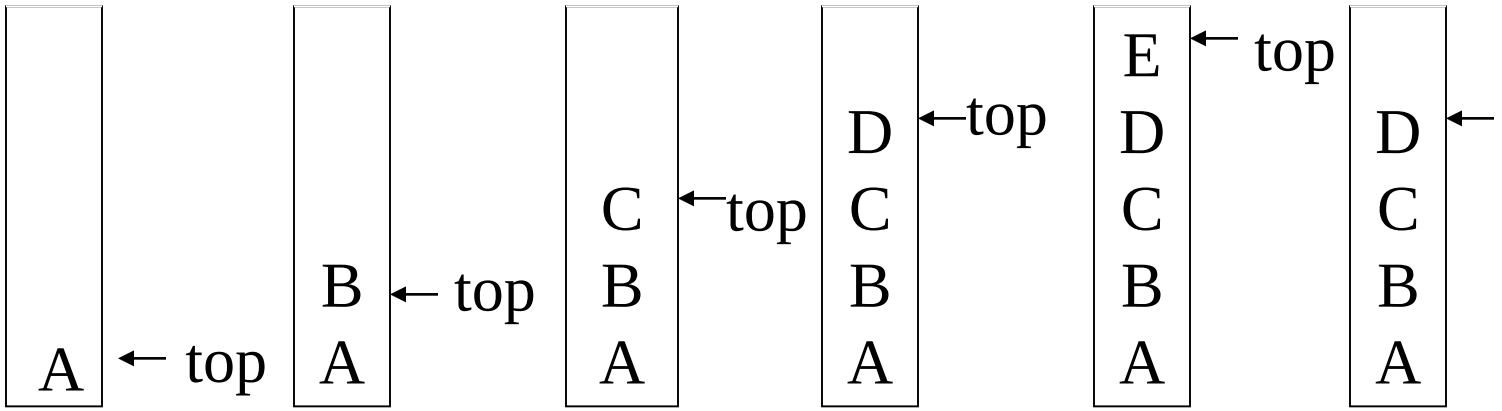
فصل سوم

پشته و صف

Stack & Queue

پشته

- پشته ساختمان داده ای است که داده ها را به ترتیب خاصی ذخیره می کند
- در پشته اخرین عضوی که وارد می شود اولین عضوی است که خارج می شود.LAST IN FIRST OUT


A
top
B
A
top
C
B
A
top
D
C
B
A
top
E
D
C
B
A
top
D
C
B
A

objects: a finite ordered list with zero or more elements.
methods:
for all *stack* ∈ *Stack*, *item* ∈ *element*, *max_stack_size* ∈ positive integer
Stack createS(*max_stack_size*) ::=
create an empty stack whose maximum size is *max_stack_size*
Boolean isFull(*stack, max_stack_size*) ::=
if (number of elements in *stack* == *max_stack_size*)
return TRUE
else return FALSE
Stack push(*stack, item*) ::=
if (IsFull(*stack*)) *stack_full*
else insert *item* into top of *stack* and **return**
Boolean isEmpty(*stack*) ::=
if(*stack* == CreateS(*max_stack_size*))
return TRUE
else return FALSE
Element pop(*stack*) ::=
if(IsEmpty(*stack*)) **return**
else remove and return the *item* on the top of the stack.

پیاده سازی پشته ها

با چند روش می توان آنها را پیاده سازی کرد:

- آرایه ها
- لیستهای پیوندی

پیاده سازی پشته ها – نمایش داده ها

```
    private:
        int top;
        KeyType *stack;
        int MaxSize;
```

```
template <class KeyType>
Stack<KeyType>::Stack (int MaxStackSize):MaxSize (MaxStackSize)
{
    stack = new KeyType[MaxSize];
    top = -1;
}
```

پیاده سازی اعمال روی پشته ها

```
template <class KeyType>
inline Boolean  Stack<KeyType>::IsFull()
{
    if (top == MaxSize -1) return TRUE;
    else return FALSE;
}

template <class KeyType>
inline Boolean  Stack<KeyType>::IsEmpty()
{
    if (top == -1) return TRUE;
    else return FALSE;
}
```

پیاده سازی اعمال روی پشته ها

```
template <class KeyType>
void Stack<KeyType>::Add (const KeyType& x)
// add x to the stack
{
    if (IsFull()) StackFull();
    else stack[++top] = x;
}
```

```
template <class KeyType>
KeyType* Stack<KeyType>::Delete (KeyType& x)
// remove and return top element from stack
{
    if (IsEmpty()) {StackEmpty(); return 0;}
    x = stack[top--];
    return &x;
}
```