线性表

2023年7月11日　　16:53

什么是线性表

多项式表示问题的启示：

1. 同一个问题可以有不同的表示（存储）方法
2. 有一类共性问题：有序线性序列的组织和管理

“线性表(Linear List)”：由同类型数据元素构成有序序列的线性结构

- 表中元素个数称为线性表的长度
- 线性表没有元素时，称为空表
- 表起始位置称表头，表结束位置称表尾

屏幕剪辑的捕获时间: 2023/7/11 16:53

线性表的抽象数据类型描述

类型名称：线性表（List）

数据对象集：线性表是 n (≥0)个元素构成的有序序列($a_1, a_2, \ldots, a_n$)

操作集：线性表L ∈ List，整数i表示位置，元素X ∈ ElementType，
线性表基本操作主要有：

1、List MakeEmpty()：初始化一个空线性表L；
2、ElementType FindKth(int K, List L)：根据位序K，返回相应元素；
3、int Find(ElementType X, List L)：在线性表L中查找X的第一次出现位置；
4、void Insert(ElementType X, int i, List L)：在位序i前插入一个新元素X；
5、void Delete(int i, List L)：删除指定位序i的元素；
6、int Length(List L)：返回线性表L的长度n。

屏幕剪辑的捕获时间: 2023/7/11 16:55

广义表

2023年7月11日　17:14

广义表

[例] 我们知道了一元多项式的表示，那么二元多项式又该如何表示？

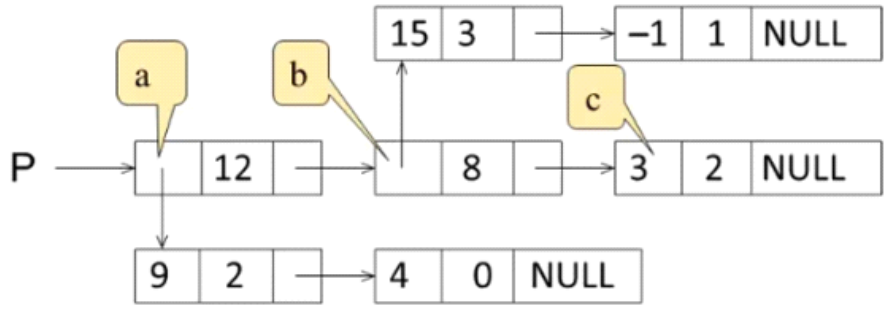
比如，给定二元多项式：$P(x,y)=9x^{12}y^2+4x^{12}+15x^8y^3-x^8y+3x^2$

【分析】可以将上述二元多项式看成关于x的一元多项式

$$P(x,y)=(9y^2+4)x^{12}+(15y^3-y)x^8+3x^2$$

$$ax^{12}+bx^8+cx^2$$

所以，上述二元多项式可以用“复杂”链表表示为：



屏幕剪辑的捕获时间: 2023/7/11 17:15

广义表(Generalized List)

- 广义表是线性表的推广
- 对于线性表而言，n个元素都是基本的单元素；
- 广义表中，这些元素不仅可以是单元素也可以是另一个广义表。

```
typedef struct GNode *GList;
struct GNode{
    int Tag;            /*标志域：0表示结点是单元素，1表示结点是广义表 */
    union {             /*子表指针域Sublist与单元素数据域Data复用，即共用存储空间*/
      ElementType   Data;
      GList  SubList;
    } URegion;
    GList Next;         /* 指向后继结点 */
};
```

Tag	Data / SubList	Next

屏幕剪辑的捕获时间: 2023/7/11 17:17

通过一个标记Tag，以及联合union，来决定这个节点是指针域还是数据域

多重链表

多重链表：链表中的节点可能同时隶属于多个链

- 多重链表中结点的指针域会有多个，如前面例子包含了Next和SubList两个指针域；
- 但包含两个指针域的链表并不一定是多重链表，比如在双向链表不是多重链表。

- 多重链表有广泛的用途：
  基本上如树、图这样相对复杂的数据结构都可以采用多重链表方式实现存储。

屏幕剪辑的捕获时间: 2023/7/11 17:34

Ex：稀疏矩阵的表示方法（十字链表）

[例] 矩阵可以用二维数组表示，但二维数组表示有两个缺陷：

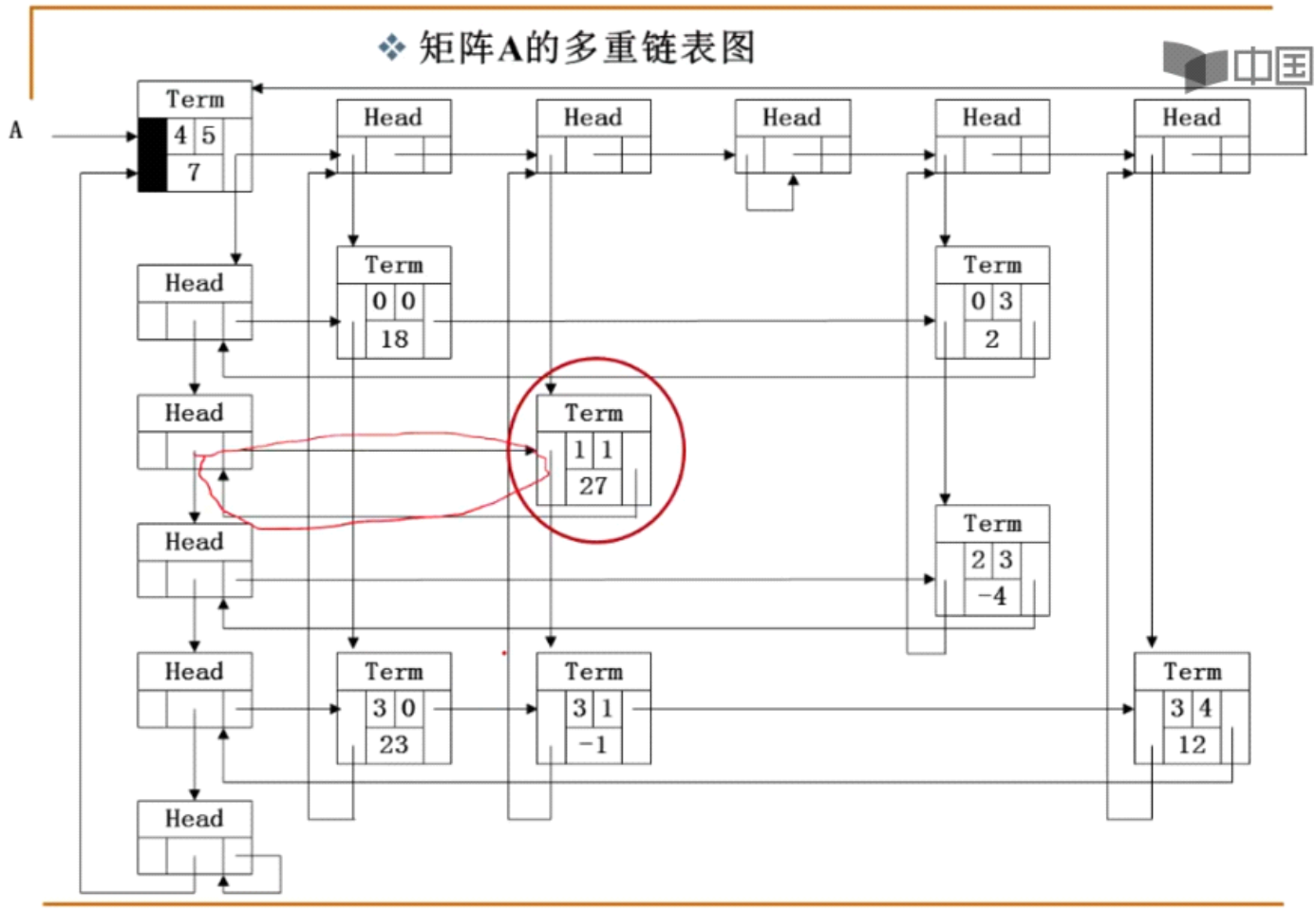
- 一是数组的大小需要事先确定，
- 对于“稀疏矩阵”，将造成大量的存储空间浪费。

$$A=\begin{bmatrix}18 & 0 & 0 & 2 & 0\\ 0 & 27 & 0 & 0 & 0\\ 0 & 0 & 0 & -4 & 0\\ 23 & -1 & 0 & 0 & 12\end{bmatrix} \qquad B=\begin{bmatrix}0 & 2 & 11 & 0 & 0 & 0\\ 3 & -4 & -1 & 0 & 0 & 0\\ 0 & 0 & 0 & 9 & 13 & 0\\ 0 & -2 & 0 & 0 & 10 & 7\\ 6 & 0 & 0 & 5 & 0 & 0\end{bmatrix}$$

【分析】采用一种典型的多重链表——十字链表来存储稀疏矩阵

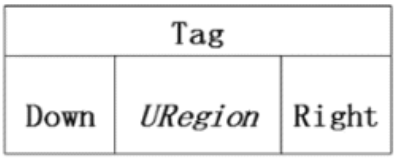
- 只存储矩阵非0元素项
  结点的数据域：行坐标Row、列坐标Col、数值Value
- 每个结点通过两个指针域，把同行、同列串起来；
  - 行指针(或称为向右指针)Right
  - 列指针（或称为向下指针）Down

多重链表表示图：

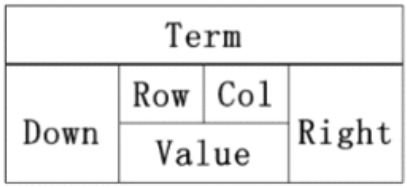


1.图中矩阵的每一行，每一列都是一个循环链表；
2.左上角的入口节点是矩阵的入口，表示矩阵行、列、非零元素数量；
3.同行列的头节点是同一个节点（例如第二行的头节点与第二列的头节点是同一个）；
4.入口节点与头节点的内部细节：

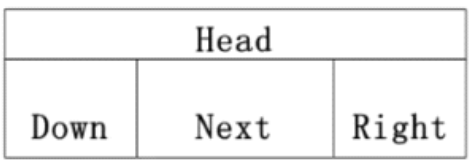
- 用一个标识域Tag来区分头结点和非0元素结点：
- 头节点的标识值为“Head”，矩阵非0元素结点的标识值为“Term”。



(a) 结点的总体结构



(b) 矩阵非0元素结点



(c) 头结点

两种节点构造上有共性，可以通过Tag来区分节点，通过union来使用同一空间内不同的数据。

堆栈、队列

2023年7月11日　17:53

1.后缀运算符：

熟悉的中缀运算符，如：a+b*c-d/e，在遇到运算符后不能直接对两边数字作运算，还要等后续的输入

上式变为后缀表达式为：abc*+de/-

变为前缀表达式为：　　-+a*bc/de

后缀运算符是读取到运算符后，立即对最近的两个数字作该运算，前缀运算符正好相反

可见**先放进表达式的数据被后拿出来**，于是就有了堆栈的概念：

后缀表达式

- 中缀表达式：运算符号位于两个运算数之间。如，$a+b*c-d/e$
- 后缀表达式：运算符号位于两个运算数之后。如，$abc*+de/-$

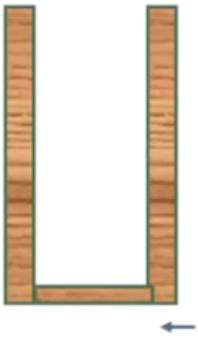
〖例〗$62/3-42*+=?$

后缀表达式求值策略：从左向右“扫描”，逐个处理运算数和运算符号

1. 遇到运算数怎么办？如何“记住”目前还不未参与运算的数？
2. 遇到运算符号怎么办？对应的运算数是什么？

启示： 需要有种存储方法，能顺序存储运算数，并在需要时“倒序”输出！

〖例〗$62/3-42*+=?$　　8



对象：6 (运算数)	对象：2 (运算数)
对象：/ (运算符)	对象：3 (运算数)
对象：– (运算符)	对象：4 (运算数)
对象：2 (运算数)	对象：* (运算符)
对象：+ (运算符)	Pop：8

堆栈的抽象数据类型描述

堆栈（Stack）：具有一定操作约束的线性表

- 只在一端（栈顶，Top）做插入、删除

- 插入数据：入栈（Push）
- 删除数据：出栈（Pop）
- 后入先出：Last In First Out（LIFO）

堆栈的抽象数据类型描述

类型名称：堆栈（Stack）

数据对象集：一个有0个或多个元素的有穷线性表。

操作集：长度为MaxSize的堆栈S ∈ Stack，堆栈元素item ∈ ElementType

1、Stack CreateStack(int MaxSize)：生成空堆栈，其最大长度为MaxSize；

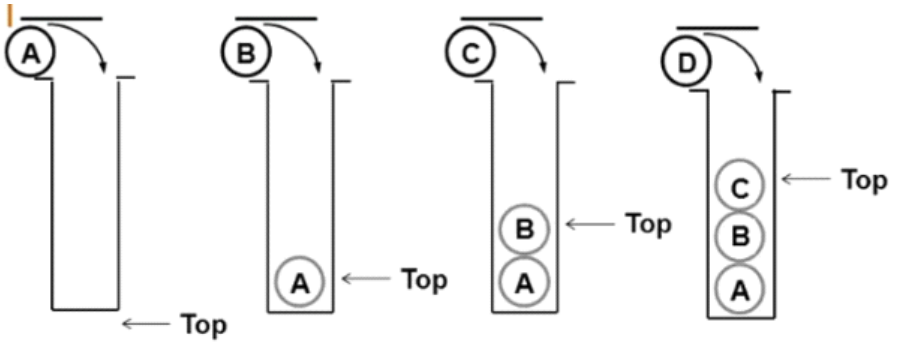
2、int IsFull(Stack S, int MaxSize)：判断堆栈S是否已满；

3、void Push(Stack S, ElementType item)：将元素item压入堆栈；

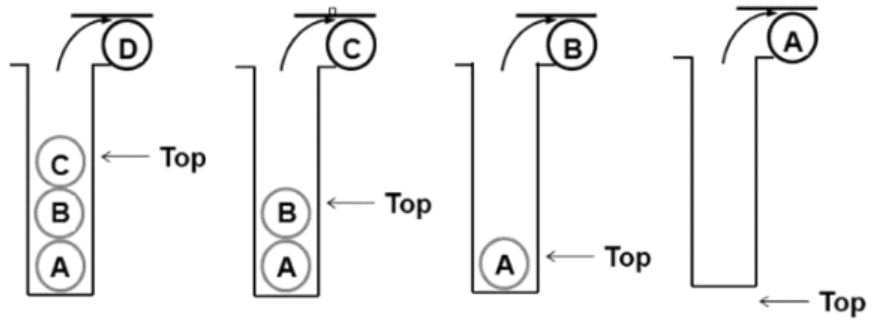
4、int IsEmpty (Stack S)：判断堆栈S是否为空；

5、ElementType Pop(Stack S)：删除并返回栈顶元素；

其中Push与Pop最为关键，做每一步时要注意堆栈是否已满



CreatStack( ); Push(S,A); Push(S,B); Push(S,C);



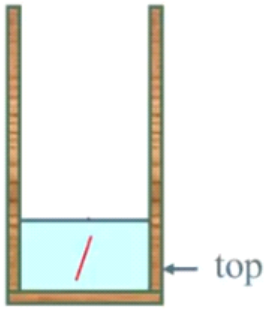
x=Pop(S); x=Pop(S); x=Pop(S); IsEmpty (S)

用单向链表表示堆栈时，如果栈顶放在尾部，当取出元素时，无法定位到上一个位置，所以应该把栈顶放在头节点后面，即创建一个头节点，头节点后面的第一个节点开始才是元素。

堆栈实现后缀表达式：

〖例〗 $a*(b+c)/d = ?$ $a\ b\ c + *\ d\ /$

输出： $a\ \ b\ \ c\ \ +\ \ *\ \ d$



输入对象: a (操作数)	输入对象: $*$ (乘法)
输入对象: $($ (左括号)	输入对象: b (操作数)
输入对象: $+$ (加法)	输入对象: c (操作数)
输入对象: $)$ (右括号)	输入对象: $/$ (除法)
输入对象: d (操作数)	

中缀表达式如何转换为后缀表达式

➢ 从头到尾读取中缀表达式的每个对象，对不同对象按不同的情况处理。

① 运算数：直接输出；
② 左括号：压入堆栈；
③ 右括号：将栈顶的运算符弹出并输出，直到遇到左括号（出栈，不输出）；
④ 运算符：
- 若优先级大于栈顶运算符时，则把它压栈；
- 若优先级小于等于栈顶运算符时，将栈顶运算符弹出并输出；再比较新的栈顶运算符，直到该运算符大于栈顶运算符优先级为止，然后将该运算符压栈；

⑤ 若各对象处理完毕，则把堆栈中存留的运算符一并输出。

❖中缀转换为后缀示例：（2*（9+6/3-5）+4）

步骤	待处理表达式	堆栈状态（底←→顶）	输出状态
1	2*（9+6/3-5）+4		
2	*（9+6/3-5）+4		2
3	（9+6/3-5）+4	*	2
4	9+6/3-5）+4	*（	2
5	+6/3-5）+4	*（	2 9
6	6/3-5）+4	*（+	2 9
7	/3-5）+4	*（+	2 9 6
8	3-5）+4	*（+/	2 9 6
9	-5）+4	*（+/	2 9 6 3
10	5）+4	*（-	2 9 6 3 / +
11	）+4	*（-	2 9 6 3 / + 5
12	+4	*	2 9 6 3 / + 5 -
13	4	+	2 9 6 3 / + 5 - *
14		+	2 9 6 3 / + 5 - * 4
15			2 9 6 3 / + 5 - * 4 +

堆栈还可以用于函数的调用（比如函数的嵌套调用）与递归实现
还有回溯算法、深度优先搜索等

队列就是先进先出，比如顺环队列。