Chapter4 数组、串与广义表

Sparse Matrix：稀疏矩阵　String：字符串　GenList：广义表

4.1 稀疏矩阵

设矩阵 A 中有 s 个非零元素，若 s 远远小于矩阵元素的总数（即s≤m×n），则称 A 为**稀疏矩阵。**

（1）设矩阵 A 中有 s 个非零元素。令 e = s/(m*n),称 e 为矩阵的稀疏因子（通常认为 e≤0.05 时称之为稀疏矩阵）

（2）在存储稀疏矩阵时，为节省存储空间，应只存储非零元素。但由于非零元素的分布一般没有规律，故在存储非零元素时，必须记下它所在的行和列的位置 (i, j)。

（3）每一个三元组 (i, j, a_{ij}) 唯一确定了矩阵A的一个非零元素。因此，稀疏矩阵可由表示非零元的一系列三元组及其行列数唯一确定。

```
template<class E>
struct Triple {              //三元组
    int row, col;            //非零元素行号/列号
    E value;                 //非零元素的值
    void operator = (Triple<E>& R)     //赋值
      { row = R.row;  col = R.col;  value = R.value; }
};

public:
    SparseMatrix (int Rw = drows, int Cl = dcols,
        int Tm = dterms);              //构造函数
    void Transpose(SparseMatrix<E>& b);   //转置
    void Add (SparseMatrix<E>& a,
        SparseMatrix<E>& b);           //a = a+b
    void Multiply (SparseMatrix<E>& a,
        SparseMatrix<E>& b);           //a = a*b
pvivate:
    int Rows, Cols, Terms;       //行 / 列 / 非零元素数
    Triple<E> *smArray;          //三元组表
};
```

矩阵的转置：

- **为加速转置速度，建立辅助数组 rowSize 和 rowStart：**
  - **rowSize记录矩阵转置前各列，即转置矩阵各行非零元素个数；**
  - **rowStart记录各行非零元素在转置三元组表中开始存放位置。**
- **扫描矩阵三元组表，根据某项列号，确定它转置后的行号，查 rowStart 表，按查到的位置直接将该项存入转置三元组表中。**

	[0]	[1]	[2]	[3]	[4]	[5]	[6]	语 义
rowSize	1	1	1	2	0	2	1	矩阵 A 各列非零元素个数
rowStart	0	1	2	3	5	5	7	矩阵 B 各行开始存放位置

A三元组	(0)	(1)	(2)	(3)	(4)	(5)	(6)	(7)
行row	0	0	1	1	2	3	4	5
列col	3	6	1	5	3	5	0	2
值value	22	15	11	17	-6	39	91	28

4.2 字符串

- 字符串是 n (≥ 0) 个字符的有限序列，
  记作 S : “$c_1c_2c_3\ldots c_n$”
  其中，S 是串名字
  “$c_1c_2c_3\ldots c_n$”是串值
  c_i 是串中字符
  n 是串的长度，n = 0 称为空串。
- 例如，S = “Tsinghua University”。
- 注意：空串和空白串不同，例如 “ ” 和 “” 分别表示长度为1的空白串和长度为0的空串。

- 串中任意个连续字符组成的子序列称为该串的子串，包含子串的串相应地称为主串。
- 设A和B分别为
  A = “This is a string” B = “is”
  则 B 是 A 的子串，A 为主串。B 在 A 中出现了两次，首次出现所对应的主串位置是2（从0开始）。因此，称 B 在 A 中的位置为2。
- 特别地，空串是任意串的子串，任意串是其自身的子串。

主要注意操作符的重构（C++学习）

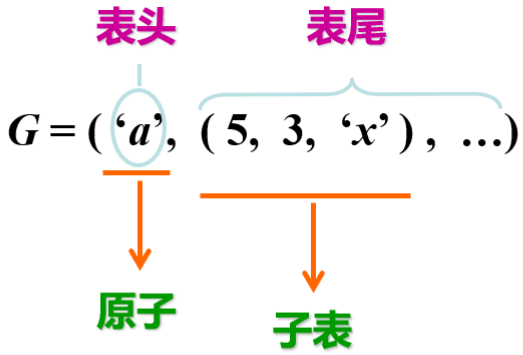
4.3 广义表

4.3.1 广义表的定义

- 广义表是 n (≥0) 个表元素组成的有限序列，记作：

$$LS(a_1, a_2, a_3, \ldots, a_n)$$

- LS 是表名，a_i 是表元素，可以是表（称为子表），可以是数据元素（称为原子）。
- n为表的长度。n = 0 的广义表为空表



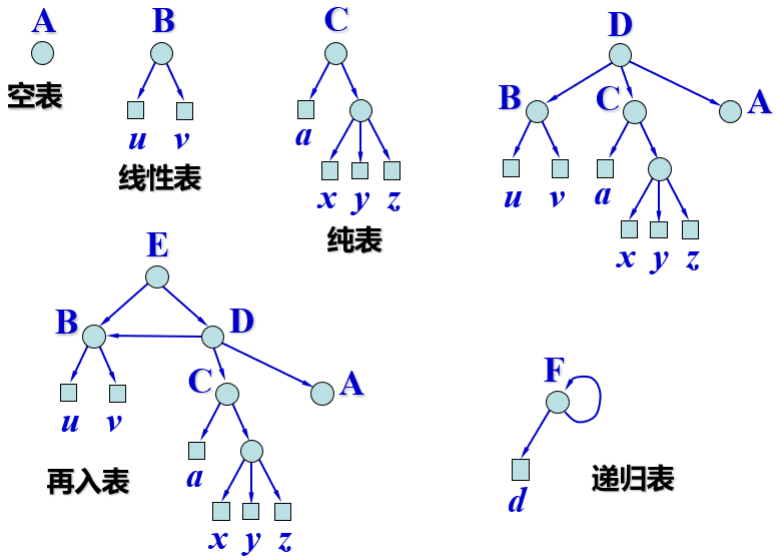
n > 0时，表的第一个表元素称为广义表 的表头（head），除此之外，其它表元素组成的表称为广义表的表尾（tail）。

广义表的第一个表元素即为该表的表头，除表头元素外其他表元素组成的**表**即为该表的表尾

表尾也是一个"表"！（要再在外圈加一个括号）

4.3.2 广义表的特性和图例

有次序性、有长度、有深度、可共享、可递归



4.3.3 广义表结点定义

- 结点类型 utype：= 0, 表头；= 1, 原子数据；= 2, 子表
- 信息info：utype = 0时, 存放引用计数(ref)；utype = 1时, 存放数据值(value)；utype = 2时, 存放指向子表表头的指针(hlink)
- 尾指针tlink：utype = 0时, 指向该表第一个结点；utype != 0时, 指向同一层下一个结点

4.3.4 广义表的递归算法

- 一个递归算法有两种：一个是递归函数的外部调用；另一个是递归函数的内部调用。
- 通常，把外部调用设置为共有函数，把内部调用设置为私有函数。
- **广义表的复制算法**
- **求广义表深度的算法**

求广义表深度的算法

$$\text{Depth(LS)}=\begin{cases}1, & \text{当LS为空表时}\\ 0, & \text{当LS为原子时}\\ 1+\max\limits_{0\le i\le n-1}\{\text{Depth}(\alpha_i)\}, & \text{其他},n\ge 1\end{cases}$$

- **广义表的删除算法**
- **建立广义表的链表表示**

递归建立广义表链表（算法2）

若s为空串，置空广义表；若s为单字符串，建立原子结点。

否则，除去s的最外层括号，为其中的每个元素递归建立子表。

函数sever(str1, hstr1) : 分离str1中的第一个非内部逗号之前的子串赋予hstr1，str1取剩余的部分。

函数createlist(ls, s) : 从字符串s建立广义表ls