教学内容---第六章

DS/SS

1. 绪论
2. 线性表
3. 栈、队列和串
4. 数组
5. 广义表
6. 树和二叉树
7. 图
8. 动态存储管理
9. 查找
10. 内部排序
11. 外部排序
12. 文件

6.1树的逻辑结构

DS/SS

基本概念和术语

- **树**：树T是n个结点的有限集。非空树中：

  （1）有且只有一个特定的称为根(Root)的结点；

  （2）当n>1时，其余结点可分为m(m>0)个互不相交的有限集$T_1,T_2,\ldots,T_m$，其中每一个集合本身又是一棵树，并且称为根的子树(SubTree)。

- **树的结点**：一个数据元素和指向其子树的分支。
- **结点的度**：结点拥有的子树数。
- **终端结点（或叶子）**：度为0的结点。
- **非终端结点（或分支结点、或内部结点）**：度不为0的结点。
- **树的度**：树内各结点的度的最大值。

6.1树的逻辑结构（续）

DS/SS

基本概念和术语

- （结点的）孩子：结点的子树的根。
- （结点的）双亲：其孩子为该结点的结点。
- 兄弟：同一个双亲的孩子之间互称兄弟。
- （结点的）祖先：从根到该结点所经分支上的所有结点。
- （结点的）子孙：以该结点为根的子树中的任一结点。
- （结点的）层次：从根开始定义，根为第一层，根的孩子为第二层。若某结点在第L层，则其孩子在第L+1层。
- 树的深度（或高度）：树中结点的最大层次。
- 有序（无序）树：树中结点的各子树有（无）从左至右次序。
- 森林(Forest)：m棵互不相交的树的集合。

6.1树的逻辑结构（续）

DS/SS

树的性质及树的表示

树的性质：

- 递归性
- 层次性

树的表示形式：

- 树形表示
- 集合嵌套表示
- 广义表形式表示
- 凹入表示

6.1树的逻辑结构（续）

树的抽象数据类型

ADT Tree {

数据对象：D={a_i | a_i属于ElemSet, i = 1,2, … ,n, n>=0}

数据关系：R: 若D为空集，则称为空树；

若D仅含有一个数据元素，则R为空集，否则R={H},H为二元关系：

(1)在D中存在唯一的称为根的数据元素root，它在关系H下无前驱；

(2)若D-{root}不为空，则存在D-{root}的一个划分D_1,D_2,…,D_m (m>0),对任意j != k(1<=j,k<=m)有D_j与D_k交集为空，且对任意的i(1<=i<=m)，唯一存在数据元素x_i属于D_i,有<root,x_i>属于H;

(3)对应于D-{root}的划分，H – {<root,x_1>,…, <root,x_m>}有唯一的一个划分H_1,H_2,…,H_m(m>0),对任意j != k(1<=j,k<=m)有H_j与H_k交集为空，且对任意的 i(1<=i<=m)， H_i 是D_i上的二元关系,(D_i,{H_i })是一棵符合本定义的树，称为根root的子树。

基本操作：P: InitTree(&T); DestroyTree(&T); …; TraverseTree(T,Visit())

}ADT Tree

6.2树的存储结构

双亲表示法

```
//----------树的双亲存储表示------------
#define MAX_TREE_SIZE          100
typedef  struct  PTNode{
        TElemType       data;      //树结点数据域
        int        parent           //双亲位置域
} PTNode;
typedef  struct {
        PTNode           nodes[MAX_TREE_SIZE ];
        int        n;        //结点数
}PTree;
```

树的双亲存储法易于实现对上层结点的操作；不易于访问底层结点。

6.2树的存储结构（续）

DS/SS

孩子表示法

便于涉及孩子操作的实现

```
typedef  struct  CTNode{
        int        child;               //孩子位置域
        struct  CTNode    *next      } *ChildPtr;
typedef  struct {
        TElemType        data;      //树结点数据域
         ChildPtr          firstchild; //孩子链表头指针  } CTBox;
typedef  struct {
        CTBox            nodes[MAX_TREE_SIZE ];
        int        n,r;      //结点数和根的位置
}PTree;
```

6.2树的存储结构（续）

DS/SS

孩子双亲表示法

```
typedef  struct  CTNode{
        int          child;                //孩子位置域
        struct  CTNode    *next      } *ChildPtr;
typedef  struct {
        int          parent                //双亲位置域
        TElemType          data;     //树结点数据域
         ChildPtr          firstchild; //孩子链表头指针  } CTBox;
typedef  struct {
        CTBox              nodes[MAX_TREE_SIZE ];
        int          n,r;              //结点数和根的位置  }PTree;
```

6.2树的存储结构

多叉链表法

结点

data	link1	link2	…	Linkk

多叉链表存储的、度为k的、结点数为n的树中，空指针域个数为：

$$n*k-(n-1) = n(k-1)+1$$

```
//----------用结构指针描述------------
typedef  struct  MultiLNode{
        ElemType          data        //数据域
        struct MultiLNode           *link1    //第1个指针域
        struct MultiLNode           *link2    //第2个指针域
        …….
        struct MultiLNode           *linkk    //第k个指针域
} MultiLNode, *MultiLinkTree
```

6.2树的存储结构（续）

DS/SS

孩子兄弟表示法（二叉链表）

结点

data	firstchild	nextsibling

```
//----------树的二叉链表（孩子-兄弟）存储表示------------
typedef  struct  CSNode{
        ElemType          data;      //树结点数据域
        struct CSNode     *firstchild,  *nextsibling;
} CSNode,    *CSTree;
```

6.2树的存储结构（续）

三叉链表

结点

data	firstchild	nextsibling	parent

```
//----------树的三叉链表（双亲-孩子-兄弟）存储表示------
typedef  struct  PCSNode{
        ElemType          data;      //树结点数据域
        struct PCSNode   *firstchild,  *nextsibling,   *parent;
} PCSNode,    *PCSTree;
```

DS/SS

二叉树的描述性定义及其基本形态

二叉树：二叉树BT是n个结点的有限集。非空二叉树中：

(1) 有且只有一个特定的称为根(Root)的结点；

(2) 当n>1时，其余结点可分为2个互不相交的有限集 BT_L,BT_R, BT_L,BT_R分别又是一棵二叉树，BT_L称为根的**左子树**；BT_R称为根的**右子树**。

二叉树的基本形态：

五种：空二叉树；仅有根结点的二叉树；仅有左子树的二叉树；仅有右子树的二叉树；左、右子树均不为空的二叉树。

6.3二叉树的逻辑结构（续）

DS/SS

ADT BinaryTree {

数据对象：D={a_i | a_i属于ElemSet, i = 1,2, … ,n, n>=0}

数据关系：R: 若D为空集，则R为空集，称为空二叉树；

若D不为空集，R={H},H为二元关系：

(1)在D中存在唯一的称为根的数据元素root，它在关系H下无前驱；

(2)若D-{root}不为空，则存在D-{root}={D_L,D_R} ,且D_L与D_R交集为空；

(3)若D_L不为空，则D_L 中存在唯一的元素x_L, <root,x_L>属于H，且存在D_L上的关系H_L 包含在H中；若D_R不为空，则D_R中存在唯一的元素x_R, <root,x_R>属于H，且存在D_R上的关系H_R 包含在H中;

H={<root,x_L>, <root,x_R >, H_L , H_R };

(4)(D_L , {H_L})是一棵符合本定义的二叉树，称为根的左子树；

(D_R , {H_R})是一棵符合本定义的二叉树，称为根的右子树。

基本操作： InitBiTree(&T); DestroyBiTree(&T); …;InsertChild (T,p,LR,c); PreOrderTraverse (T,Visit()); InOrderTraverse (T,Visit()); PostOrderTraverse (T,Visit()); LevelOrderTraverse (T,Visit())

}ADT BinaryTree

6.3二叉树的逻辑结构（续）

DS/SS

二叉树的数学性质

- 性质1：在二叉树的第i层上至多有2^{i-1}个结点(i>=1);
- 性质2：深度为k的二叉树至多有2^k-1个结点(k>=1);
- 性质3：二叉树T，若叶子结点数为n_0,度为2的结点数为n_2，则有$n_0 = n_2 + 1$;

性质1：

[证明]

i=1时，显然有$2^{i-1} = 2^0 = 1$；

假设对于所有的j，1<=j<i，第j层上至多有2^{j-1}个结点，则当j=i时，由假设知2^{i-1}层上至多有2^{i-2},而二叉树的每个结点的度至多为2，故在第i层上最大结点数为i-1层上最大结点数的2倍，即$2* 2^{i-2} = 2^{i-1}$。 [证毕]

6.3二叉树的逻辑结构（续）

DS/SS

二叉树的数学性质

性质3：

[证明]

$n = n_0 + n_1 + n_2$

$n = B+1$

$B = n_1 + 2n_2$

所以： $n_0 = n_2 + 1$ [证毕]

6.3二叉树的逻辑结构（续）

DS/SS

二叉树的数学性质

- 满二叉树：深度为k且有2^k-1个结点的二叉树;
- 完全二叉树：深度为k，有n个结点的二叉树，当且仅当其每一个结点都与深度为k的满二叉树中编号从1至n的结点一一对应时，称之为完全二叉树。
- 性质4：具有n个结点的完全二叉树的深度为：以2为底的n的对数值取下整+1；
- 性质5：n个结点的完全二叉树结点按照层序编号，则对任一结点i(1<=i<=n)，有：（1）如果i=1，则结点i是二叉树的根；如果i>1，则其双亲是结点(i/2)取下整；(2)如果2i>n,则结点i无左孩子（结点i为叶子结点）；否则其左孩子是结点 2i；(3)如果2i+1>n,则结点i无右孩子；否则其右孩子是结点 2i+1。

6.3二叉树的逻辑结构（续）

DS/SS

二叉树的数学性质

性质4：

[证明]

假设深度为k，由性质2以及完全二叉树定义有：

$2^{k-1}-1 < n <= 2^k-1$ 推出 $2^{k-1} <= n < 2^k$

$k-1 <=$ 以2为底的n的对数 $< k$ 由于k是整数，所以结论成立。 [证毕]