

数据结构与算法 第4章 树 Slide. 4 - 1

第4章 树

两个ADT：
 ADT树
 ADT二元树

ADT

- 定义
 - 逻辑结构及其特征
 - 基本操作（算法）
- 实现
 - 存储结构（描述）
 - 存储结构特点
 - 存储结构的定义
 - 操作（算法）实现
 - 算法的性能分析
- 应用

数据结构的核心算法：遍历算法

哈尔滨工业大学 计算机科学与技术学院 张岩

数据结构与算法 第4章 树 Slide. 4 - 2

第4章 树

4.1 基本术语
 4.2 二元树
 4.3 树
 4.4 森林与二元树间的转换
 4.5 树的应用

哈尔滨工业大学 计算机科学与技术学院 张岩

数据结构与算法 第4章 树 Slide. 4 - 3

4.1 基本术语

[定义] 1、一个结点 x 组成的集{ x }是一株树，这个结点 x 也是这株树的根。
 2、假设 x 是一个结点， T_1, T_2, \dots, T_k 是 k 株互不相交的树，我们可以构造一株新树：令 x 为根，并有 k 条边由 x 指向树 T_1, T_2, \dots, T_k 。这些边也叫做分支， T_1, T_2, \dots, T_k 称作根 x 的树之子树（SubTree）。

树的构造性递归定义

分支 分支 分支
 子树 子树 子树

注意！
 ■ 递归定义，但不会产生循环定义。
 ■ 一株树的每个结点都是这株树的某株子树的根。

哈尔滨工业大学 计算机科学与技术学院 张岩

数据结构与算法 第4章 树 Slide. 4 - 4

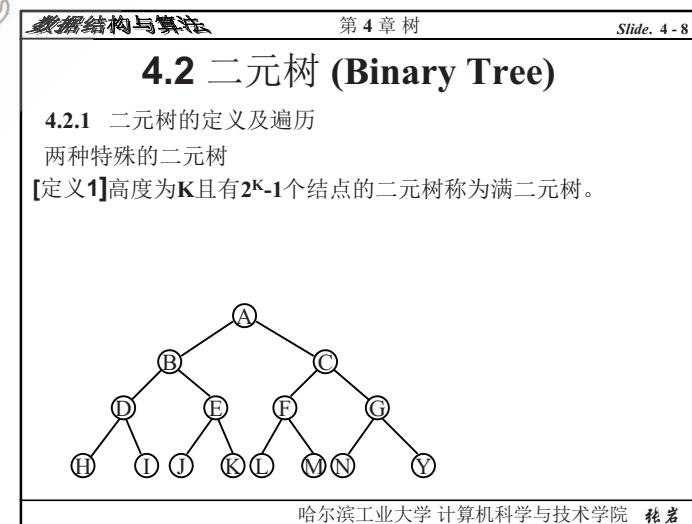
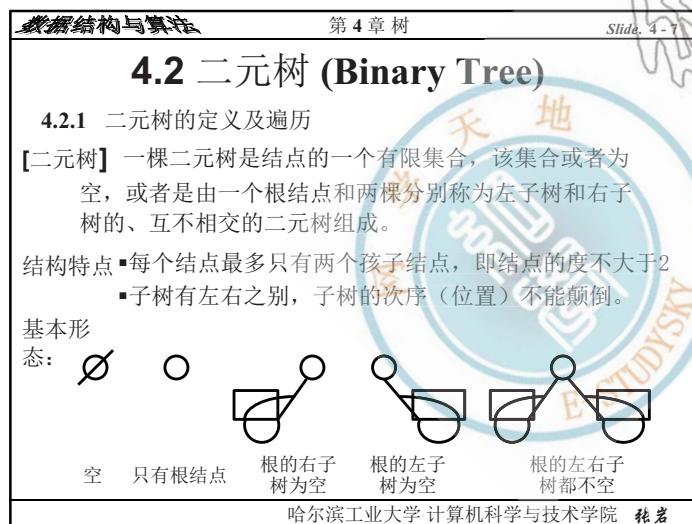
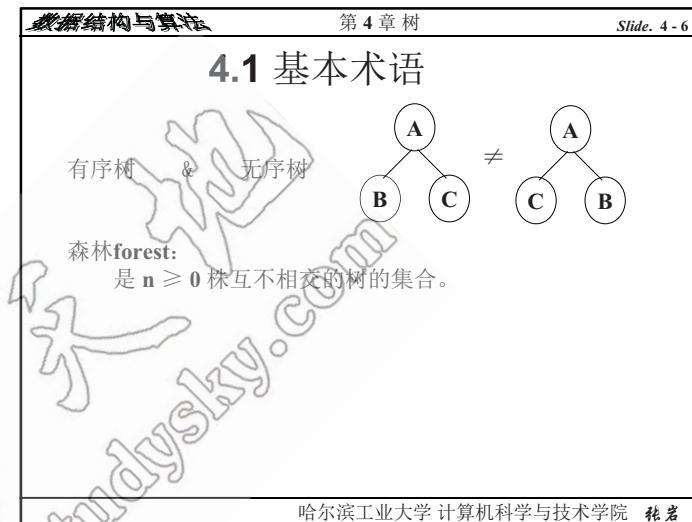
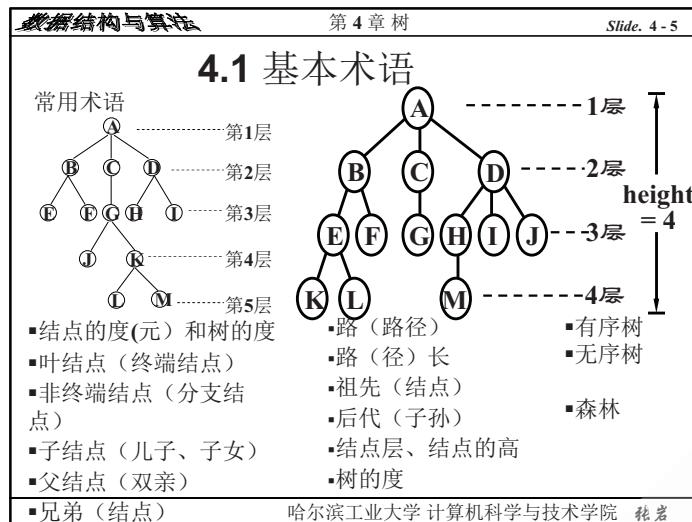
4.1 基本术语

1层
 2层
 3层
 4层
 5层
 height = 4

树的逻辑结构特点

除根结点之外，每株子树的根结点有且仅有一个直接前驱，但可以有0个或多个直接后继。即一对多的关系，反映了结点之间的层次关系。

哈尔滨工业大学 计算机科学与技术学院 张岩



4.2 二元树 (Binary Tree)

4.2.1 二元树的定义及遍历

两种特殊的二元树

[定义2]设二元树高度为K，称满足下列性质的二元树为完全二元树

- (1) 所有的叶都出现在K或K-1层；
- (2) K-1层的所有叶都在非终结结点的右边；
- (3) 除了K-1层的最右非终结结点可能有一个（只能是左分支）或两个分支之外，其余非终结结点都有两个分支。

哈尔滨工业大学 计算机科学与技术学院 张岩

4.2 二元树 (Binary Tree)

4.2.1 二元树的定义及遍历

[二元树的遍历]

根据某种策略，按照一定的顺序访问二元树中的每一个结点，使每个结点被访问一次且只被访问一次。结果得到二元树结点的线性序列。

根(D)、左孩子(L)和右孩子(R)三个结点可能出现的顺序有：

- ① DLR
- ② LDR
- ③ LRD
- ④ DRL
- ⑤ RDL
- ⑥ RLD

策略：左孩子结点一定要在右孩子结点之前访问。

要讨论的三种操作分别为：

①先根顺序DLR， ②中根顺序LDR， ③后根顺序LRD

哈尔滨工业大学 计算机科学与技术学院 张岩

4.2.1 二元树的遍历的递归定义】

- ①先根顺序遍历二元树：
若二元树非空则：
 {
 访问根结点；
 先根顺序遍历左子树；
 先根顺序遍历右子树；
 };
- ②中根顺序遍历二元树：
若二元树非空则：
 {
 中根顺序遍历左子树；
 访问根结点；
 中根顺序遍历右子树；
 };
- ③后根顺序遍历二元树：
若二元树非空则：
 {
 后根顺序遍历左子树；
 后根顺序遍历右子树；
 访问根结点；
 };

所得到的线性序列分别称为先根(序)、中根(序)和后根(序)序列。

哈尔滨工业大学 计算机科学与技术学院 张岩

4.2.2 二元树的性质

性质1 二元树的第*i*层最多有 2^{i-1} 个结点。 $(i \geq 1)$
[证明用数学归纳法]

性质2 高度为K的二元树最多有 $2^K - 1$ 个结点。 $(K \geq 1)$
[证明用求等比级数前K项和的公式]
 $2^0 + 2^1 + 2^2 + \dots + 2^K = 2^K - 1$

性质3 对任何一棵二元树，如果其叶结点有 n_0 个，度为2的非叶结点有 n_2 个，则有 $n_0 = n_2 + 1$

证明：若设度为1的结点有 n_1 个，结点总数为n，分支总数为B，则根据二元树的定义，

$$n = n_0 + n_1 + n_2 \quad B = 2n_2 + n_1 = n - 1$$

因此，有 $2n_2 + n_1 = n_0 + n_1 + n_2 - 1$

$$n_2 = n_0 - 1 \implies n_0 = n_2 + 1$$

哈尔滨工业大学 计算机科学与技术学院 张岩

哈工大计算机考研全套视频和资料，真题、考点、典型题、命题规律独家视频讲解！
详见：网学天地（www.e-studysky.com）；咨询QQ：2696670126

数据结构与算法		第4章 树	Slide. 4 - 13
性质4 具有 n ($n \geq 0$) 个结点的完全二元树的高度为 $\lceil \log_2(n+1) \rceil$ 或 $\lfloor \log_2 n \rfloor + 1$			
证明：设完全二元树的高度为 K , 则有			
$2^{K-1} - 1 < n \leq 2^K - 1$ <p style="text-align: center;">前 $k-1$ 层 第 k 层</p>			
变形 $2^{K-1} < n + 1 \leq 2^K$			
取对数 $K-1 < \log_2(n+1) \leq K$			
因此有 $\lceil \log_2(n+1) \rceil = K$			
哈尔滨工业大学 计算机科学与技术学院 张岩			

数据结构与算法		第4章 树	Slide. 4 - 14																				
性质5 完全(或满)二元树的顺序存储结构及其性质																							
如果将一棵有 n 个结点的完全二元树自顶向下, 同一层自左向右连续给结点编号, 1, 2, ..., n , 且使该编号对应于数组的下标, 则有以下关系:																							
<ul style="list-style-type: none"> □ 若 $i = 1$, 则 i 是根结点, 无父结点 □ 若 $i > 1$, 则 i 的父结点为 $\lfloor i/2 \rfloor$ □ 若 $2*i \leq n$, 则 i 有左儿子且为 $2*i$; 否则, i 无左儿子。 □ 若 $2*i+1 \leq n$, 则 i 有右儿子且为 $2*i+1$; 否则, i 无右儿子。 □ 若 i 为偶数, 且 $i \leq n$, 则有右兄弟, 且为 $i+1$。 □ 若 i 为奇数, 且 $i \leq n \&& i != 1$, 则其左兄弟, 且为 $i-1$ 																							
<table border="1" style="margin-left: auto; margin-right: auto;"> <tr> <td>A</td><td>B</td><td>C</td><td>D</td><td>E</td><td>F</td><td>G</td><td>H</td><td>I</td><td>J</td> </tr> <tr> <td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td><td>10</td> </tr> </table>				A	B	C	D	E	F	G	H	I	J	1	2	3	4	5	6	7	8	9	10
A	B	C	D	E	F	G	H	I	J														
1	2	3	4	5	6	7	8	9	10														
哈尔滨工业大学 计算机科学与技术学院 张岩																							

数据结构与算法		第4章 树	Slide. 4 - 15
4.2.3 抽象数据型二元树			
操作: ① EMPTY (BT) ;			
② ISEMPTY (BT) ;			
③ CREATEBT (V, LT, RT) ;			
④ LCHILD (BT) ;			
⑤ RCHILD (BT) ;			
⑥ DATA (BT) ;			
<pre>void PREORDER (BT) BTREE BT; { if (!ISEMPTY (BT)) { visit (DATA (BT)); PREORDER (LCHILD (BT)); PREORDER (RCHILD (BT)); } }</pre>			
例1-1: 写一个递归函数, 按先根顺序列出二元树中每个结点的 DATA 域之值。			
<pre>{ if (!ISEMPTY (BT)) { visit (DATA (BT)); PREORDER (LCHILD (BT)); PREORDER (RCHILD (BT)); } }</pre>			
哈尔滨工业大学 计算机科学与技术学院 张岩			

数据结构与算法		第4章 树	Slide. 4 - 16
<pre>void INORDER (BT) BTREE BT; { if (!ISEMPTY (BT)) { INORDER (LCHILD (BT)); visit (DATA (BT)); INORDER (RCHILD (BT)); } }</pre>			
例1-2: 写一个递归函数, 按中根顺序列出二元树中每个结点的 DATA 域之值。			
<pre>{ if (!ISEMPTY (BT)) { INORDER (LCHILD (BT)); visit (DATA (BT)); INORDER (RCHILD (BT)); } }</pre>			
<pre>void POSTORDER (BT) BTREE BT; { if (!ISEMPTY (BT)) { POSTORDER (LCHILD (BT)); POSTORDER (RCHILD (BT)); visit (DATA (BT)); } }</pre>			
例1-3: 写一个递归函数, 按后根顺序列出二元树中每个结点的 DATA 域之值。			
<pre>{ if (!ISEMPTY (BT)) { POSTORDER (LCHILD (BT)); POSTORDER (RCHILD (BT)); visit (DATA (BT)); } }</pre>			
哈尔滨工业大学 计算机科学与技术学院 张岩			

哈工大计算机考研全套视频和资料，真题、考点、典型题、命题规律独家视频讲解！
详见：网学天地（www.e-studysky.com）；咨询QQ：2696670126

数据结构与算法 第4章 树 Slide. 4 - 17

二元树的遍历的非递归过程

例

```

void INORDER ( BT )
BTREE BT ;
{ if ( !ISEMPTY ( BT ) )
  { INORDER ( LCHILD ( BT ) );
    visit ( DATA ( BT ) );
    INORDER ( RCHILD ( BT ) )
  }
}
先序: A B D J H E C F I G K L M
中序: J D H B E A F I C G L K M
后序: J H D E B I F L M K G C A
      27 ^ # 结束
  
```

数据结构：
设栈S：
用以保留当前结点；

算法：
Loop：
{
 if (T 非空)
 {进栈；左走一步；}
 else
 {退栈；右走一步；}
};

数据结构与算法 第4章 树 Slide. 4 - 18

二元树的遍历的非递归过程

```

void NINORDER( BT )
BTREE BT;
{ STACK S ; BTREE T ;
  MAKENULL( S );
  T = BT ;
  while ( !ISEMPTY( T ) || !EMPTY( S ) )
    if ( !ISEMPTY( T ) )
      { PUSH( T , S );
        T = T → LCHILD ( T );
      }
    else
      { T = TOP ( S );
        POP ( S );
        visit( DATA(T) );
        T = T → RCHILD ( T );
      }
}
  
```

哈尔滨工业大学 计算机科学与技术学院 张岩

数据结构与算法 第4章 树 Slide. 4 - 19

4.2.4 二元树的表示

一、顺序表示

1、完全（或满）二元树

根据性质5，如已知某结点的层序编号i，则可求得该结点的双亲结点、左孩子结点和右孩子结点。

采用一维数组，按层序顺序依次存储二元树的每一个结点。如下图所示：

A	B	C	D	E	F	G	H	I	J
1	2	3	4	5	6	7	8	9	10

哈尔滨工业大学 计算机科学与技术学院 张岩

数据结构与算法 第4章 树 Slide. 4 - 20

2、一般二元树

通过虚设部分结点，使其变成相应的完全二元树。

根据性质5，如已知某结点的层序编号i，则可求得该结点的双亲结点、左孩子结点和右孩子结点，然后检测其值是否为虚设的特殊结点\$。

一株n个结点的二元树，最坏情况下需要 $2^n - 1$ 个数组单元。

哈尔滨工业大学 计算机科学与技术学院 张岩

哈工大计算机考研全套视频和资料，真题、考点、典型题、命题规律独家视频讲解！
详见：网学天地（www.e-studysky.com）；咨询QQ：2696670126

数据结构与算法 第4章 树 Slide. 4 - 21

二、左右链表示

```

lchild | data | rchild
struct node {
    struct node *lchild;
    struct node *rchild;
    datatype data;
};

typedef struct node *BTREE;

例: (P102)
BTREE CREATEBT(v, ltree, rtree)
datatype v; BTREE ltree, rtree;
{ BTREE root;
    root = new node;
    root->data = v; root->lchild = ltree; root->rchild = rtree;
    return root;
}

```

哈尔滨工业大学 计算机科学与技术学院 张岩

数据结构与算法 第4章 树 Slide. 4 - 22

例1: 按先序序列建立二元树的左右链结构。
如由图所示二元树输入:

ABDH##I##E##CFJ##G##

其中:#表示空

```

void CreateBtree(BTREE &T)
{
    cin >> ch;
    if (ch == '#') T = NULL;
    else { if (!(T = new BTREE)) exit(1);
        T->data = ch;
        CreateBtree(T->lchild);
        CreateBtree(T->rchild);
    }
}

```

哈尔滨工业大学 计算机科学与技术学院 张岩

数据结构与算法 第4章 树 Slide. 4 - 23

例2: 建立二元树的左右链结构的非递归算法

```

struct node *s[max]; /* 辅助指针数组, 存放二元树结点指针 */
BTREE CREATBT()
{ int i, j; datatype ch;
    struct node *bt, *p; /* bt为根, p 用于建立结点 */
    cin >> i >> ch;
    while (i != 0 && ch != 0){
        p = new node; p->data = ch;
        p->lchild = NULL; p->rchild = NULL;
        s[i] = p;
        if (i == 1) bt = p;
        else { j = i / 2; /* 父结点的编号 */
            if (i % 2 == 0) s[j]->lchild = p; /* i是j的左儿子 */
            else s[j]->rchild = p; /* i是j的右儿子 */
            cin >> i >> ch;
        }
    }
}

```

哈尔滨工业大学 计算机科学与技术学院 张岩

数据结构与算法 第4章 树 Slide. 4 - 24

例3 计算二元树结点个数的递归算法

```

struct node {
    struct node *lchild;
    struct node *rchild;
    datatype data;
};

typedef node *BTREE;

例4 求二元树高度的递归算法
int Height(BTREE T)
{
    if (T == NULL) return 0;
    else return 1 + Height(T->lchild) + Height(T->rchild);
}

例5 删除二元树的递归算法
void destroy(BTREE T)
{
    if (T != NULL) {
        destroy(T->lchild);
        destroy(T->rchild);
        delete T;
    }
}

```

哈尔滨工业大学 计算机科学与技术学院 张岩

哈工大计算机考研全套视频和资料，真题、考点、典型题、命题规律独家视频讲解！
详见：网学天地（www.e-studysky.com）；咨询QQ：2696670126

数据结构与算法 第4章 树 Slide. 4 - 25

例6 利用栈实现先序遍历的非递归算法—栈顶保存当前结点的右子树

```
void PreOrder( BTREE T )
{ STACK S; MAKENULL(S); //递归工作栈
  struct node* p = T; PUSH( S, NULL );
  while ( p != NULL ) {
    cout << p->data << endl;
    if ( p->rchild != NULL )
      PUSH( S, p->rchild );
    if ( p->lchild != NULL )
      p = p->lchild; //进左子树
    else POP(S, p); //左子树空, 进右子树 {}
  }
  访问 A B D H I E C F G
  进栈 C E I * * * G J
  P左进 B D H - - - F
  退栈 + + + I E C + G
}
```

哈尔滨工业大学 计算机科学与技术学院 张岩

数据结构与算法 第4章 树 Slide. 4 - 26

例7 阅读以下二元树操作算法说明其功能。

```
void unknown( BTREE T )
{ struct *p = T, *tmp;
  if ( p != NULL ) {
    temp = p->lchild;
    p->lchild = p->rchild;
    p->rchild = temp;
    unknown( p->lchild );
    unknown( p->rchild );
  }
  struct node {
    struct node *lchild;
    struct node *rchild;
    datatype data;
  };
  typedef node * BTREE;
}
```

使用栈消去递归算法中的两个递归语句

```
void unknown( BREE T )
{struct node *p, *temp;
STACK S; MAKENULL (S);
if (T != NULL) {
  PUSH(T, S);
  while ( ! EMPTY(S) ) {
    POP(S, p); //栈中退出一个结点
    temp = p->lchild; //交换子女
    p->lchild = p->rchild;
    p->rchild = temp;
    if (p->rchild != NULL)
      PUSH( p->rchild , S );
    if (p->lchild != NULL)
      PUSH (S, p->lchild );
  }
}
```

哈尔滨工业大学 计算机科学与技术学院 张岩

数据结构与算法 第4章 树 Slide. 4 - 27

三、二元树的其它表示

1、三叉链表结点结构

```
struct node {
  struct node *lchild;
  struct node *rchild;
  struct node *parent;
  datatype data;
};

typedef node * TriTREE;
```

哈尔滨工业大学 计算机科学与技术学院 张岩

数据结构与算法 第4章 树 Slide. 4 - 28

root

二元树

root

二叉链表

root

三叉链表

二元树链式表示的示例

哈尔滨工业大学 计算机科学与技术学院 张岩

哈工大计算机考研全套视频和资料，真题、考点、典型题、命题规律独家视频讲解！
详见：网学天地（www.e-studysky.com）；咨询QQ：2696670126

数据结构与算法 第4章 树 Slide. 4 - 29

2、静态二叉、三叉链表表示

	data	parent	lchild	rchild
0	A	-1	1	-1
1	B	0	2	3
2	C	1	-1	-1
3	D	1	4	5
4	E	3	-1	-1
5	F	3	-1	-1

哈尔滨工业大学 计算机科学与技术学院 张岩

数据结构与算法 第4章 树 Slide. 4 - 30

四、二元树的线索表示——线索二元树
问题的提出：

- 在n个结点的二元树左右链表示中，有n+1个空链域。如何利用n+1个空链域，使二元树的操作更加方便。
- 在二元树左右链表示中，为求某个结点的（中序）前驱p->ltag或（中序）后继p->rtag，每次都要从树根开始进行查找，很不方便。

定义：

- 若结点p有左孩子，则p->lchild指向其左孩子结点，否则令其指向其（先序、中序、后序）前驱；
- 若结点p有右孩子，则p->rchild指向其右孩子结点，否则令其指向其（先序、中序、后序）后继；
- 同时在每个结点中增加两个标志位，以区分该结点的两个链域是指向其左/右孩子还是指向某种遍历的前驱/后继。

哈尔滨工业大学 计算机科学与技术学院 张岩

数据结构与算法 第4章 树 Slide. 4 - 31

结点类型

```

enum bool {FALSE, TRUE};

struct node {
    datatype data;
    struct node *lchild, *rchild;
    enum bool lchild, rchild;
};

typedef struct node * THTREE;

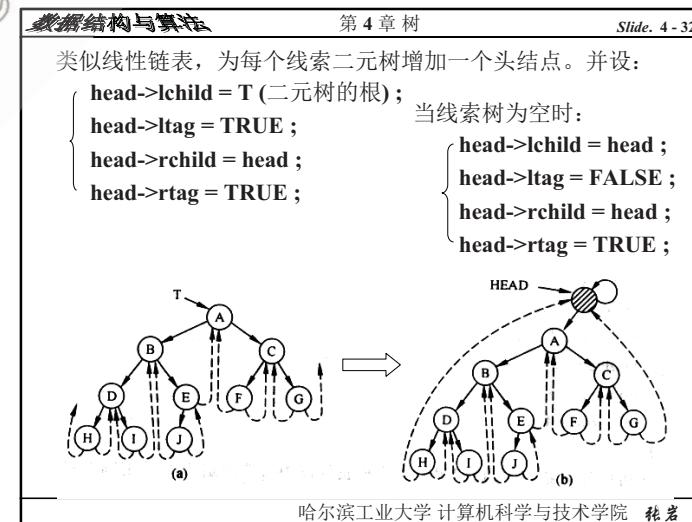
p->ltag = {
    TRUE   p->lchild 指向左孩子
    FALSE  p->lchild 指向（中序）前驱
};

p->rtag = {
    TRUE   p->rchild 指向右孩子
    FALSE  p->lchild 指向（中序）后继
};

```

讨论：为方便操作利用了 n+1 个结点，但为实现操作却多用了 2n 个标志位。（代价不大）

哈尔滨工业大学 计算机科学与技术学院 张岩



哈工大计算机考研全套视频和资料，真题、考点、典型题、命题规律独家视频讲解！
详见：网学天地（www.e-studysky.com）；咨询QQ：2696670126

数据结构与算法 第4章 树 Slide. 4 - 33

线索二元树的若干算法

算法1：求p\$ (中序后继)：

分析：(1) 当 $p \rightarrow rtag == \text{FALSE}$ 时， $p \rightarrow rchild$ 即为所求 (线索)。
 (2) 当 $p \rightarrow rtag == \text{TRUE}$ 时， $\text{INNEXT}(\text{THTREE } p)$

```

THTREE INNEXT(THTREE p)
{
    THTREE Q;
    Q=p->rchild;
    if (p->rtag == TRUE)
        while(Q->ltag == TRUE)
            Q = Q->lchild;
    return (Q);
}
    
```

哈尔滨工业大学 计算机科学与技术学院 张岩

数据结构与算法 第4章 树 Slide. 4 - 34

算法2：求\$p (中序前驱)：

分析：(1) 当 $p \rightarrow ltag == \text{FALSE}$ 时， $p \rightarrow lchild$ 即为所求 (线索)。
 (2) 当 $p \rightarrow ltag == \text{TRUE}$ 时，\$p 为 p 的左子树的最右结点。

```

THTREE INPRE(THTREE p)
{
    THTREE Q;
    Q=p->lchild;
    if (p->ltag == TRUE)
        while(Q->rtag == TRUE)
            Q = Q->rchild;
    return (Q);
}
    
```

哈尔滨工业大学 计算机科学与技术学院 张岩

数据结构与算法 第4章 树 Slide. 4 - 35

算法3：利用INNEXT算法，中序遍历线索二元树。

```

void THINORDER(THTREE HEAD)
{
    THTREE temp;
    temp = HEAD;
    do {
        temp = INNEXT (temp);
        if (temp != HEAD)
            visit (temp->data);
    } while (temp != HEAD);
}
    
```

哈尔滨工业大学 计算机科学与技术学院 张岩

数据结构与算法 第4章 树 Slide. 4 - 36

算法4：中序线索二元树的插入算法。

在二元树中一般不讨论结点的插入与删除，原因是其插入与删除的操作与线性表相同，所不同的是需要详细说明操作的具体要求。

而在线索树中结点的插入与删除则不同，因为要同时考虑修正线索的操作。

例：将结点 R 插入作为结点 S 的右孩子结点。

- (1) 若 S 的右子树为空，插入比较简单；
- (2) 若 S 的右子树非空，则 R 插入后 原来 S 的右子树作为 R 的右子树

哈尔滨工业大学 计算机科学与技术学院 张岩

哈工大计算机考研全套视频和资料，真题、考点、典型题、命题规律独家视频讲解！
详见：网学天地（www.e-studysky.com）；咨询QQ：2696670126

数据结构与算法 第4章 树 Slide. 4 - 37

```
void RINSERT ( THTREE S, THTREE R )
{ THTREE W;
  R->rchild = S->rchild ;
  R->rtag = S->rtag ;
  R->lchild = S ;//-
  R->ltag = FALSE ;//-
  S->rchild = R ;
  S->rtag = TRUE ;
  if ( R->rtag == TRUE )
    { w = INNEXT( R );
      w->lchild = R ;
    }
}
```

哈尔滨工业大学 计算机科学与技术学院 张岩

数据结构与算法 第4章 树 Slide. 4 - 38

算法5：求中序线索二元树中结点p先序顺序的后继结点p*的算法
分析：(1) p 的左子树不空时, p 的左儿子p->lchild 即为 p*;
(2) p 的左子树空但右子树不空时, p 的p->rchild 为p*;
(3) p 的左右子树均空时, 右线索序列中的一个有右孩子结点的右儿子或头结点

```
THTREE PRENEXT(THTREE p)
{
  THTREE Q;
  if (p->ltag == TRUE)
    Q=p->lchild;
  else{ Q = p;
    while(Q->rtag == FALSE)
      Q = Q->rchild;
    Q = Q->rchild;
  } return (Q);
}
```

哈尔滨工业大学 计算机科学与技术学院 张岩

数据结构与算法 第4章 树 Slide. 4 - 39

算法6：二元树的（中序）线索化算法-----递归算法
基本思想：将二元树变为线索二元树的过程称为二元树的线索化。按某种次序将二元树线索化，只要按某种次序遍历二元树，在遍历过程中用线索取代空指针。为此，附设一个指针pre--始终指向刚刚访问过的结点，而指针p指示当前正在访问的结点。显然，结点*pre是结点*p的前驱，而*p是结点*pre的后继中序线索化算法要点：

- 若结点*p有空指针域，则将相应的标志位置为FALSE表明是线索；
- 若结点*p中有中序前驱结点*pre(即pre!=NULL)，则
 - 若结点*pre的右线索标志已建立（即pre->rtag==FASLE），则令pre->rchild为指向其中序后继结点*p的右线索；
 - 若结点*p的左线索标志已建立（即p->ltag==FASLE），则令p->lchild为指向其中序前驱结点*pre的左线索；
- 将pre指向刚刚访问过得结点*p（即pre = p）。这样，在下次访问一个新结点*p时，*pre为其前驱结点。

BTREE *p=NULL; //全局量

```
void InorderTh(BTREE *p) //将二元树 p 中序线索化
{ if( p ) //p 非空时, 当前访问的结点是 p
  InorderTh( p->lchild ); //左子树线索化
  p->ltag=( p->lchild ) ? TRUE : FALSE; //左 (右) 儿子非空
  p->rtag=( p->rchild ) ? TRUE : FALSE; //右, 标志1; 否: 0
  if ( pre ) //若*p 的前驱*pre 存在
    if ( pre->rtag == FASLE ) // *p 的前驱右标志为线索
      pre->rchild=p; //令 *pre 的右线索指向中序后继
    if ( p->ltag == FASLE ) // *p 的左标志为线索
      p->lchild=pre; //令 *p 的左线索指向中序前驱
  }
  pre = p; //令 pre 是下一个访问的中序前驱
  InorderTh( p->rchild ); //右子树线索化
}
```

哈尔滨工业大学 计算机科学与技术学院 张岩

数据结构与算法 第4章 树 Slide. 4 - 40

```
BTREE *p=NULL; //全局量
void InorderTh(BTREE *p) //将二元树 p 中序线索化
{ if( p ) //p 非空时, 当前访问的结点是 p
  InorderTh( p->lchild ); //左子树线索化
  p->ltag=( p->lchild ) ? TRUE : FALSE; //左 (右) 儿子非空
  p->rtag=( p->rchild ) ? TRUE : FALSE; //右, 标志1; 否: 0
  if ( pre ) //若*p 的前驱*pre 存在
    if ( pre->rtag == FASLE ) // *p 的前驱右标志为线索
      pre->rchild=p; //令 *pre 的右线索指向中序后继
    if ( p->ltag == FASLE ) // *p 的左标志为线索
      p->lchild=pre; //令 *p 的左线索指向中序前驱
  }
  pre = p; //令 pre 是下一个访问的中序前驱
  InorderTh( p->rchild ); //右子树线索化
}
```

哈尔滨工业大学 计算机科学与技术学院 张岩

哈工大计算机考研全套视频和资料，真题、考点、典型题、命题规律独家视频讲解！
详见：网学天地（www.e-studysky.com）；咨询QQ：2696670126

数据结构与算法	第4章 树	Slide. 4 - 41
<p>4.2.5 二元树的复制</p> <p>二元树的相似与等价</p> <p>两株二元树具有相同结构指： 形状相同</p> <p>(1) 它们都是空的；</p> <p>(2) 它们都是非空的，且左右子树分别具有相同结构。 义具有相同结构的二元树为相似二元树。</p> <p>相似且相应结点包含相同信息的二元树称为等价二元树。</p>		

哈尔滨工业大学 计算机科学与技术学院 张岩

数据结构与算法	第4章 树	Slide. 4 - 42
<pre> 判断两株二元树是否等价 int EQUAL(BTREE firstbt, BTREE secondbt) { int x; x = 0; if (ISEMPTY(firstbt) && ISEMPTY(secondbt)) x = 1; else if (!ISEMPTY(firstbt) && !ISEMPTY(secondbt)) if (DATA(firstbt) == DATA(secondbt)) if (EQUAL(LCHILD(firstbt), LCHILD(secondbt))) x = EQUAL(RCHILD(firstbt), RCHILD(secondbt)); return(x); } /* EQUAL */ </pre>		

哈尔滨工业大学 计算机科学与技术学院 张岩

数据结构与算法	第4章 树	Slide. 4 - 43
<p>二元树的复制</p> <pre> BTREE COPY(BTREE oldtree) { BTREE temp; if (oldtree != NULL) { temp = new Node; temp->lchild = COPY(oldtree->lchild); temp->rchild = COPY(oldtree->rchild); temp->data = oldtree->data; return (temp); } return (NULL); } /* COPY */ </pre>		

哈尔滨工业大学 计算机科学与技术学院 张岩

数据结构与算法	第4章 树	Slide. 4 - 44
<h3>4.3 树</h3> <h4>4.3.1 抽象数据型树</h4> <ul style="list-style-type: none"> ➤ PARENT(n , T) 求结点 n 的双亲 ➤ LEFTMOST_CHILD(n , T) 返回结点 n 的最左儿子 ➤ RIGHT_SIBLING(n , T) 返回结点 n 的右兄弟 ➤ DATA(n , T) 返回结点 n 的信息 ➤ CREATE_k(v , T_1 , T_2 , , T_k) , k = 1 , 2 , <p>建立data域v的根结点r，有k株子树T₁，T₂，.....，T_k，且自左至右排列；返回r。</p> <p>➤ ROOT(T) 返回树T的根结点</p>		

哈尔滨工业大学 计算机科学与技术学院 张岩

哈工大计算机考研全套视频和资料，真题、考点、典型题、命题规律独家视频讲解！
详见：网学天地（www.e-studysky.com）；咨询QQ：2696670126

数据结构与算法 第4章 树 Slide. 4 - 45

树的三种遍历

- 先根顺序
 - ◇访问根结点：
 - 先根顺序遍历 T_1 ；
 - 先根顺序遍历 T_2 ；
 - ...
 - 先根顺序遍历 T_k ；
- 先根遍历序列：RADEBCFGHK
中根遍历序列：DAERBGFKHC
后根遍历序列：DEABGHKFCR
- 中根顺序
- 后根顺序
 - 中根顺序遍历 T_1 ；
 - 访问根结点；
 - 中根顺序遍历 T_2 ；
 - ...
 - 中根顺序遍历 T_k ；
 - 访问根结点；

哈尔滨工业大学 计算机科学与技术学院 张岩

数据结构与算法 第4章 树 Slide. 4 - 46

例：假设树的类型为TREE，结点的类型为node，数据项的类型为elementtype，用递归方法给出树的先根遍历如下：

```
void PREORDER(node n, TREE T)
{ node c;
  visit(DATA(T));
  c = LEFTMOST-CHILD(n, T);
  while (c != NULL)
    PREORDER(c, T);
    c = RIGHT-SIBLING(c, T);
}
```

先根遍历整棵树：PREORDER(ROOT(T), T)

思考题：写出树的中根遍历和后根遍历算法

哈尔滨工业大学 计算机科学与技术学院 张岩

数据结构与算法 第4章 树 Slide. 4 - 47

4.3.2 树的存储结构

1、树的数组表示法（双亲表示、父链表示、单链表示）

父链表示：

首先，对树T的结点按下列规则依次编号：(与完全二元树不同)

- (1) 根结点编号为1；
- (2) 对于T中的每一个已编号的结点k，按从左到右的顺序对k的儿子结点编号。

然后，令树T的结点编号与一个结构体数组的下标对应，结构体数组的每个单元包括两个域：parent和data域，且规定：

$A[i].parent = \begin{cases} j & \text{若结点 } i \text{ 的父亲是 } j \\ 0 & \text{若结点 } i \text{ 是根} \end{cases}$

$A[i].data = \text{结点 } i \text{ 的数据域的值}$

T	0	1	1	2	2	5	5	5	3	3	parent
	A	B	C	D	E	H	I	J	F	G	data
0	1	2	3	4	5	6	7	8	9	10	

哈尔滨工业大学 计算机科学与技术学院 张岩

数据结构与算法 第4章 树 Slide. 4 - 48

结构特点：

- 每个结点均保存父结点所在的数组单元下标
- 兄弟结点的编号连续。

存储结构：

```
struct node {
  int parent;
  char data;
};

typedef node TREE[11];
TREE T;
```

基本操作的实现：

T	0	1	1	2	2	5	5	5	3	3	parent
	A	B	C	D	E	H	I	J	F	G	data
0	1	2	3	4	5	6	7	8	9	10	

- 易求父结点、祖先结点：如i的父结点的父结点 $T[T[i].parent].parent$
- 易求结点数据域的值： $T[i].data$
- 不便于CREATE操作

哈尔滨工业大学 计算机科学与技术学院 张岩

哈工大计算机考研全套视频和资料，真题、考点、典型题、命题规律独家视频讲解！
详见：网学天地（www.e-studysky.com）；咨询QQ：2696670126

数据结构与算法 第4章 树 Slide. 4 - 49

基本操作的实现：

```

node LEFTMOST-CHILD( node n , TREE T )
{ node i;
  for ( i = n + 1 ; i <= maxnodes -1 ; i++)
    if ( T[i].parent == n ) // i为n的最左儿子
      return( i ); // i为最左孩子
    return( 0 ); // n是叶子
}

node RIGHT-SIBLING ( node n , TREE T )
{ node i = n + 1 ;
  if ( T[i].parent == T[n].parent)
    return( i ); // i为 n的有兄弟
  return( 0 ); // n 没有右兄弟
}

```

哈尔滨工业大学 计算机科学与技术学院 张岩

数据结构与算法 第4章 树 Slide. 4 - 50

2、树的邻接表表示法（孩子表示法、孩子链表表示法）

```

typedef struct CTNode {
  int child;
  struct CTNode *next; } *ChildPtr;
typedef struct {
  Telementtype data;
  ChildPtr firstchild; } CTBox;
typedef struct {
  CTBox nodes[MAX_TREE_SIZE];

```

(a)

(b)

哈尔滨工业大学 计算机科学与技术学院 张岩

数据结构与算法 第4章 树 Slide. 4 - 51

3、树的（左）孩子（右）兄弟表示法（二元树表示法）

1、结点结构

firstchild	data	nexstsibling
------------	------	--------------

2、存储示例

因每个结点有且仅有两个指针域，所以也称为二叉链或二元树表示方法

哈尔滨工业大学 计算机科学与技术学院 张岩

数据结构与算法 第4章 树 Slide. 4 - 52

3、树的（左）孩子（右）兄弟表示法（二元树表示法）
类型: **typedef struct CSNode {/动态存储结构**

```

  ElemtType data;
  struct CSNode *firstchild , *nexstsibling;
} ;
typedef struct CSNode *TREE;

```

遍历 | 树 | 二元树

先根	RADEBCFGHK	RADEBCFGHK
中根	DAERBGHFKC	DEABGHKFCR
后根	DEABGHKFRC	EDKHGFCBAR

哈尔滨工业大学 计算机科学与技术学院 张岩

哈工大计算机考研全套视频和资料，真题、考点、典型题、命题规律独家视频讲解！
详见：网学天地（www.e-studysky.com）；咨询QQ：2696670126

数据结构与算法		第4章 树	Slide. 4 - 53
4.4 森林与二元树			
一、森林(树)转换为二元树：			
<ul style="list-style-type: none"> ■ 连线：把每株树的各兄弟结点连起来； 把各株树的根结点连起来（视为兄弟） ■ 抹线：对于每个结点，只保留与其最左儿子的连线，抹去该结点与其它结点之间的连线 ■ 旋转：按顺时针旋转45度角（左链竖画，右链横画） 			
哈尔滨工业大学 计算机科学与技术学院 张岩			

数据结构与算法		第4章 树	Slide. 4 - 54
二、二元树转换为森林(树)：			
<ul style="list-style-type: none"> ■ 连线：若某个结点 k 是其双亲结点的左孩子，则将该结点 k 的有孩子以及（当且仅当）连续地沿着右孩子的右链不断搜索到的所有右孩子，都分别与结点 k 的双亲结点相连； ■ 抹线：把原二元树中的所有结点与其右孩子的连线以及（当且仅当）连续地沿着右孩子的右链不断搜索到的所有右孩子的连线全部抹去； ■ 旋转：按逆时针旋转45度角（即把结点按层次排列） 			
哈尔滨工业大学 计算机科学与技术学院 张岩			

数据结构与算法		第4章 树	Slide. 4 - 55
三、森林(树)与二元树之间的对应关系：			
<ul style="list-style-type: none"> ■ 将一株树转换为二元树，二元树一定没有右子树（原因？） ■ 一般结论：森林中的任何没有右兄弟的结点在对应的二元树中，该没有右子树； ■ 任何一个森林(树)对应唯一的一株二元树，反之亦然。 ■ 且第一株树的根对应二元树的根；第一株树的所有子树森林对应二元树的左子树；其余子树森林对应二元树的右子树； 			
四、森林转换为二元树的形式化描述（递归算法）			
输入： $F = \{T_1, T_2, \dots, T_n\}$ 输出： $\text{二元树 } B(F)$ 算法：p117			
五、二元树转换为森林的形式化描述（递归算法）			
<ul style="list-style-type: none"> ■ 若 B 为空，则 F 为空； ■ 若 B 不空，则 <ul style="list-style-type: none"> ■ F 中的第一株树 T_1 的根对应二元树 B 的根； ■ T_1 中根结点的子树森林 F_1 是由 B 的左子树转换来的； ■ F 中除 T_1 之外其余子树组成的森林 $F' = \{T_2, \dots, T_n\}$ 是由 B 的右子树转换而来的。 			
哈尔滨工业大学 计算机科学与技术学院 张岩			

数据结构与算法		第4章 树	Slide. 4 - 56
六、非空森林的基本遍历顺序			
<ul style="list-style-type: none"> ■ 先根顺序 访问第一株树的根结点； 按先根顺序遍历第一株树的子树森林； 按先根顺序遍历其余子树森林。 ■ 后根顺序 按后根顺序遍历第一株树的子树森林； 访问第一株树的根结点； 按后根顺序遍历其余子树森林。 			
先根顺序：ABCDEFJG 后根顺序：BKCAHEJFGD			
哈尔滨工业大学 计算机科学与技术学院 张岩			

哈工大计算机考研全套视频和资料，真题、考点、典型题、命题规律独家视频讲解！
详见：网学天地（www.e-studysky.com）；咨询QQ：2696670126

数据结构与算法 第4章 树 Slide. 4 - 57

七、树的（简单）应用—表达式处理（表达式求值）

1.用树结构表示表达式：

- 叶结点表示操作数；
- 非叶结点表示运算符：

二元运算符有两株子树对应于它的操作数；
一元运算符有一株子树对应于它的操作数。

后缀表示 $a - b * (c + d) / \ln(x)$
前缀表示 $- a / * b + c d \ln x$
后缀表示 $a b c d + * x \ln / -$

3. 表达式的波兰表示

- 表达式树的先根遍历序列称为该表达式的前缀表示或波兰表示
- 表达式树的后根遍历序列称为该表达式的后缀表示或逆波兰表示

哈尔滨工业大学 计算机科学与技术学院 张岩

数据结构与算法 第4章 树 Slide. 4 - 58

七、树的（简单）应用—表达式处理（表达式求值）

3.表达式求值

- 把中缀表达式转换为后缀表达式（栈结构、树结构）
- 根据后缀表达式计算表达式的值

中缀表示 $a - b * (c + d) / \ln(x)$
前缀表示 $- a / * b + c d \ln x$
后缀表示 $a b c d + * x \ln / -$

4.优点

- 不需要使用括号；
- 不用考虑运算符的优先级

中缀表示： $3 * \ln(x+1) - a / x^2$
前缀表示： $- * 3 \ln + x 1 / a ^ x 2$
后缀表示： $3 x 1 + \ln a x 2 ^ / -$

哈尔滨工业大学 计算机科学与技术学院 张岩

数据结构与算法 第4章 树 Slide. 4 - 59

4.5 树的应用

4.5.1 用树结构表示集合

一、ADT集合MFSET

- 集合：性质相同的元素所组成的整体（有限且互不相交）。
- 定义在集合上的基本操作
 - (1) UNION(S_i, S_j, S)： If $S_i \cap S_j = \emptyset$, $S = S_i \cup S_j$;
 - (2) FIND(i, S)：求包含*i*的集合；
 - (3) INITIAL(A, x)：建立集合*A*，使之只包含*x*。

例如， $S1=\{1, 7, 8, 9\}$, $S2=\{2, 5, 10\}$, $S3=\{3, 4, 6\}$
则 $S1 \cup S2 = \{1, 2, 5, 7, 8, 9, 10\}$

哈尔滨工业大学 计算机科学与技术学院 张岩

数据结构与算法 第4章 树 Slide. 4 - 60

二、ADT MFSET的实现

1. 集合的树结构表示（父链表示）

令集合的元素对应于数组的下标，而相应的元素值表示其父结点所对应的数组单元下标。

parent	0	1	2	3	4	5	6	7	8	9	10
	8	5	0	3	0	3	1	1	1	5	

数组下标：代表元素名
根结点的下标：集合名

并：把其中之一当成另一株树的子树即可。
包含：求元素所在的树根。

哈尔滨工业大学 计算机科学与技术学院 张岩

哈工大计算机考研全套视频和资料，真题、考点、典型题、命题规律独家视频讲解！
详见：网学天地（www.e-studysky.com）；咨询QQ：2696670126

数据结构与算法	第4章 树	Slide. 4 - 61
<p>2. 集合的存储结构</p> <pre>#define n 元素的个数 typedef int MFSET[n+1]; /* 集合的类型为MFSET,元素的类型为int */ 3. 基本操作的实现 void UNION(int i, int j ,MFSET parent) { parent[i]=j; /* 归并, 结果树之根为j */ } void INITIAL(int x ,MFSET parent) { parent[x]=0; }</pre>		

哈尔滨工业大学 计算机科学与技术学院 张岩

数据结构与算法	第4章 树	Slide. 4 - 62
<p>2. 集合的存储结构</p> <pre>#define n 元素的个数 typedef int MFSET[n+1]; /* 集合的类型为MFSET,元素的类型为int */ 3. 基本操作的实现 int FIND(int i, MFSET parent) { int tmp=i; while(parent[tmp]!=0)/*>0,未到根 */ tmp=parent[tmp];/* 上溯 */ return tmp; }</pre>		

哈尔滨工业大学 计算机科学与技术学院 张岩

数据结构与算法	第4章 树	Slide. 4 - 63
<p>4. 性能分析</p> <pre>UNION(1, 2, parent), FIND(1, parent) UNION(2, 3, parent), FIND(1, parent) UNION(3, 4, parent), FIND(1, parent) UNION(n-1, n, parent), FIND(1, parent)</pre> <p>每次执行UNION的时间都是O(1)，共n - 1次，所需时间O(n)；而每个FIND(1, parent)，需要从1开始找到根，当1位于第i层时，FIND(1, parent)所需时间为O(i)，共n - 2次，所需时间为O($\sum i$) = O(n^2) 原因：在并操作时，将结点多的并入结点少的，从而形成单链树。</p>		

哈尔滨工业大学 计算机科学与技术学院 张岩

数据结构与算法	第4章 树	Slide. 4 - 64
<p>三、改进的ADT MFSET的实现</p> <p>1. 基本想法：改进并操作的原则，即将结点少的并入结点多的；另相应的存储结构也要提供支持—以加权规则压缩高度。 3. 基本操作的实现</p> <p>2. 存储结构：</p> <pre>void UNION(int A,int B, MFSET C) { if(C[A].count>C[B].count){/* B < A */ C[B].father=A; /* 并入A */ C[A].count+=C[B].count; int father; int count; /* 加权 */ }MFSET[n+1]; else/* A < B */ C[A].father=B; /* 并入B */ C[B].count+=C[A].count; }</pre>		

哈尔滨工业大学 计算机科学与技术学院 张岩

哈工大计算机考研全套视频和资料，真题、考点、典型题、命题规律独家视频讲解！
详见：网学天地（www.e-studysky.com）；咨询QQ：2696670126

数据结构与算法	第4章 树	Slide. 4 - 65
3. 基本操作的实现		
<pre>int FIND(int x, MFSET C) { int tmp=x; while(C[tmp].father!=0)/*>0,未到根 */ tmp=C[tmp].father; /* 上溯 */ return tmp; } void INITIAL(int A ,MFSET C) { C[x].father=0; C[x].count=1; }</pre>		

数据结构与算法	第4章 树	Slide. 4 - 66
四、等价分类		
<p>等价关系：集合S上具有自反性、对称性和传递性的二元关系R。 等价类：$x \in S, y \in S, x \equiv y \Leftrightarrow (x,y) \in R \text{ 或 } xRy$。 集合S上的一个等价关系唯一确定一个等价类的集合S / R(商集)。 等价分类：把一个集合分成若干个等价类的过程（分清、分净） 等价分类算法： 例 集合S={1,2,3,4,5,6,7}的等价对如下： $1 \equiv 2, 5 \equiv 6, 3 \equiv 4, 1 \equiv 4$</p>		

数据结构与算法	第4章 树	Slide. 4 - 67
四、等价分类		
<p>等价分类算法： 例 集合S={1,2,3,4,5,6,7}的等价对如下： $1 \equiv 2, 5 \equiv 6, 3 \equiv 4, 1 \equiv 4$ step1:令S中的每一个元素自身构成一个等价类，S1,S2,...S7 step2: (1)重复读入等价对(i, j)； (2)对每个读入的等价对(i, j),求出i 和j 所在的集合Sk 和Sm (不失一般性) (3)若Sk ≠ Sm,则将Sk并入Sm,并将Sk置空。 当所有的等价对处理过后，S1,S2,...S7中的非空集合即为S的R等价类</p>		

数据结构与算法	第4章 树	Slide. 4 - 68
<pre>void EQUIVA(MFSET S) { int i,j,k,m; for(i=1; i<=n+1;i++) INITIAL(i,S);/*使集合S只包含元素i */ cin>>i>>j; /* 读入等价对*/ while(!(i==0&&j==0){/* 等价对未读完*/ k=FIND(i,S);/*求i的根*/ m=FIND(j,S);/* 求j的根*/ if(k!=m){/*if k==m,i,j已在一个树中，不需合并*/ UNION(i,j,S);/*合并*/ cout<<i<<j; } } }</pre>		

数据结构与算法 第4章 树 Slide. 4 - 69

4.5.2 树的应用—判定树

假定有八枚硬币 **a**、**b**、**c**、**d**、**e**、**f**、**g**、**h**，已知其中1枚是伪造的假币，假币的重量与真币不同，或重或轻。要求以天平为工具，用最少的比较次数挑出假币。

H—假币重于真币；L—假币轻于真币。 算法见P123
哈尔滨工业大学 计算机科学与技术学院 张岩

数据结构与算法 第4章 树 Slide. 4 - 70

4.5 树的应用

4.5.3 哈夫曼 (Huffman) 树及其应用

一、哈夫曼树及其构造

1. 扩充二元树 [内结点○ (增长树) → 外结点□]

在二元树中，对于每个结点，若出现空（左/右）子树，则为其加上一个特殊的结点（称为外结点），由此得到的新二元树称为原二元树的扩充二元树。而原二元树的结点称为内结点

- 没有度数为1的结点
- 外结点数=内结点数 + 1
- 有 n 个外结点的扩充二元树共有 $2n-1$ 个结点。

哈尔滨工业大学 计算机科学与技术学院 张岩

数据结构与算法 第4章 树 Slide. 4 - 71

4.5.3 哈夫曼 (Huffman) 树及其应用

一、哈夫曼树及其构造

2. 扩充二元树的外路径长、内路径长及相互关系

- 外路径长 **E**: 从根结点到每个外结点的路径长之和， $E = \sum l_j$
- 内路径长 **I**: 从根结点到每个内结点的路径长之和
- 关系: $E = I + 2 \cdot n$ (n 为内结点的个数)

内路径长 **I** = 2 1+3 2+1 3 = 11
外路径长 **E** = 1 2+5 3+2 4 = 25

哈尔滨工业大学 计算机科学与技术学院 张岩

数据结构与算法 第4章 树 Slide. 4 - 72

4.5.3 哈夫曼 (Huffman) 树及其应用

一、哈夫曼树及其构造

3. 扩充二元树的加权路径长

- 外结点的加权路径长:
设每个外结点 **j** 对应一个实数 w_j (称为外结点的权值)， l_j 是从根到外结点 **j** 的路长，则 $w_j \cdot l_j$ 称为外结点 **j** 的加权路长。
- 扩充二元树的加权路长:
 $WPL = \sum w_j \cdot l_j$ 称为扩充二元树的加权路长。

设: $w_i = \{2, 3, 4, 11\}$
求: $\sum w_j \cdot l_j$ (加权路长)

(a) 11 1+4 2+2 3+3 3=34
(b) 2 1+3 2+4 3+11 3=53
(c) 2 2+11 2+3 2+4 2=40

哈尔滨工业大学 计算机科学与技术学院 张岩

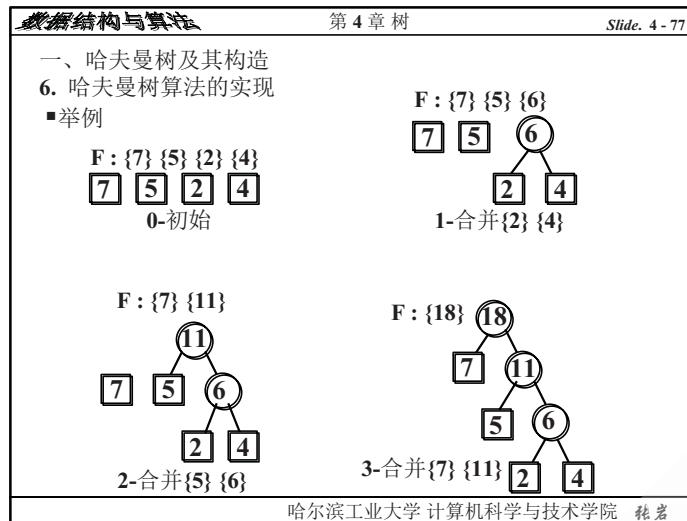
数据结构与算法		第4章 树	Slide. 4 - 73
一、哈夫曼树及其构造			
4. 哈夫曼树（最优二元树）			
在给定权值为 $w_1, w_2 \dots w_n$ 的 n 个叶结点所构成的所有扩充二元树中， $WPL = \sum w_j \cdot l_j$ 最小的称为 huffman 树。			
在哈夫曼树中，权值越小，离根越远；权值大的结点离根最近。			
5. 构造哈夫曼树的算法			
(1) 由给定的 n 个权值 $\{w_0, w_1, w_2, \dots, w_{n-1}\}$ ，构造具有 n 棵扩充二元树的森林 $F = \{T_0, T_1, T_2, \dots, T_{n-1}\}$ ，其中每棵扩充二叉树 T_i 只有一个带权值 w_i 的根结点，其左、右子树均为空。			
(2) 重复以下步骤，直到 F 中仅剩下一棵二元树为止：			
① 在 F 中选取两棵根结点的权值最小的扩充二元树，做为左、右子树构造一棵新的二元树，且置新的二元树的根结点的权值为其左、右子树上根结点的权值之和。			
② 在 F 中删去这两棵二元树。			
③ 把新的二元树加入 F 。			
哈尔滨工业大学 计算机科学与技术学院 张岩			

数据结构与算法		第4章 树	Slide. 4 - 74
一、哈夫曼树及其构造			
6. 哈夫曼树算法的实现			
■ 存储结构定义			
#define n 100 /* 叶子树 */			
#define m 2*(n)-1 /* 结点总数 */			
typedef struct /* 结点型 */			
double weight; /* 权值 */			
int lchild; /* 左孩子链 */			
int rchild; /* 右孩子链 */			
int parent; /* 双亲链 */			
} HTNODE;			
HTNODE T[m]; /* huffman 树的静态三叉链表表示 */			
HuffmanT T;			
/* huffman 树的静态三叉链表表示 */			
哈尔滨工业大学 计算机科学与技术学院 张岩			

数据结构与算法		第4章 树	Slide. 4 - 75
一、哈夫曼树及其构造			
6. 哈夫曼树算法的实现			
■ 算法要点			
1) 初始化：将 $T[0], \dots, T[m-1]$ 共 $2n-1$ 个结点的三个链域均置空 (-1)，权值为 0；			
2) 输入权值：读入 n 个叶子的权值存于 T 的前 n 个单元 $T[0], \dots, T[n]$ ，它们是 n 个独立的根结点上的权值；			
3) 合并：对森林中的二元树进行 $n-1$ 次合并，所产生的新结点依次存放在 $T[i]$ ($n <= i <= m-1$)。每次合并分两步：			
(1) 在当前森林中的二元树 $T[0], \dots, T[i-1]$ 所有结点中选取权值最小和次最小的两个根结点 $T[p_1]$ 和 $T[p_2]$ 作为合并对象，这里 $0 <= p_1, p_2 <= i-1$ ；			
(2) 将根为 $T[p_1]$ 和 $T[p_2]$ 的两株二元树作为左、右子树合并为一株新二元树，新二元树的根结点为 $T[i]$ 。即			
$T[p_1].parent = T[p_2].parent = i, T[i].lchild = p_1, T[i].rchild = p_2, T[i].weight = T[p_1].weight + T[p_2].weight.$			
哈尔滨工业大学 计算机科学与技术学院 张岩			

数据结构与算法		第4章 树	Slide. 4 - 76
一、哈夫曼树及其构造			
6. 哈夫曼树算法的实现			
■ 算法的求精			
void CreartHT(HuffmanT T) /* 构造 huffman 树，T[m-1] 为其根 */			
{ int i, p1, p2;			
InitHT(T); /* 1) 初始化 */			
InputW(T); /* 2) 输入权值 */			
for (i = n; i < m; i++) { /* 3) n-1 次合并 */			
SelectMin(T, i-1, &p1, &p2); /* 3)(1) */			
T[p1].parent = T[p2].parent = i; /* 3)(2) */			
T[i].lchild = p1;			
T[i].rchild = p2;			
T[i].weight = T[p1].weight + T[p2].weight;			
}			
}			
哈尔滨工业大学 计算机科学与技术学院 张岩			

哈工大计算机考研全套视频和资料，真题、考点、典型题、命题规律独家视频讲解！
详见：网学天地（www.e-studysky.com）；咨询QQ：2696670126



数据结构与算法 第4章 树 Slide. 4 - 78

一、哈夫曼树及其构造
6. 哈夫曼树算法的实现
■ 举例

	weight	parent	lchild	rchild
7	7	-1	-1	-1
5	5	-1	-1	-1
2	2	-1	-1	-1
4	4	-1	-1	-1
	-1	-1	-1	-1
	-1	-1	-1	-1
	-1	-1	-1	-1
	-1	-1	-1	-1

哈尔滨工业大学 计算机科学与技术学院 张岩

数据结构与算法 第4章 树 Slide. 4 - 79

一、哈夫曼树及其构造
6. 哈夫曼树算法的实现
■ 举例

	weight	parent	lchild	rchild
0	7	-1	-1	-1
1	5	-1	-1	-1
2	4	-1	-1	-1
3	4	-1	-1	-1
4	6	-1	2	3
5	-1	-1	-1	-1
6	-1	-1	-1	-1

p1 → 2
p2 → 3
i → 4

7 5 6
2 4

哈尔滨工业大学 计算机科学与技术学院 张岩

数据结构与算法 第4章 树 Slide. 4 - 80

一、哈夫曼树及其构造
6. 哈夫曼树算法的实现
■ 举例

	weight	parent	lchild	rchild
7	7	-1	-1	-1
5	5	7	-1	-1
2	2	5	-1	-1
4	4	5	-1	-1
6	6	5	2	3
11	11	-1	1	4
	-1	-1	-1	-1

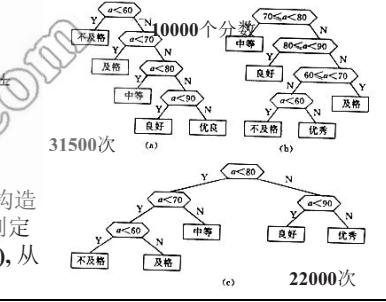
p1 → 1
p2 → 4
i → 5

哈尔滨工业大学 计算机科学与技术学院 张岩

哈工大计算机考研全套视频和资料，真题、考点、典型题、命题规律独家视频讲解！
详见：网学天地（www.e-studysky.com）；咨询QQ：2696670126

数据结构与算法 第4章 树 Slide. 4 - 81				
一、哈夫曼树及其构造 6. 哈夫曼树算法的实现 ■ 举例				
weight	parent	lchild	rchild	
7	6	-1	-1	p1 → 0
5	5	-1	-1	1
2	4	-1	-1	2
4	4	-1	-1	3
6	5	2	3	4
11	6	1	4	i → 5
18	-1	0	5	

哈尔滨工业大学 计算机科学与技术学院 张岩

数据结构与算法 第4章 树 Slide. 4 - 82																
4.5.3 哈夫曼 (Huffman) 树及其应用																
二、优化(分类统计的)判定过程																
例：输入一批学生成绩，将百分制转换成五分制。并且已知：																
<table border="1"> <thead> <tr> <th>分数</th> <th>0~59</th> <th>60~69</th> <th>70~79</th> <th>80~89</th> <th>90~100</th> </tr> </thead> <tbody> <tr> <td>比例数</td> <td>0.05</td> <td>0.15</td> <td>0.40</td> <td>0.30</td> <td>0.10</td> </tr> </tbody> </table>					分数	0~59	60~69	70~79	80~89	90~100	比例数	0.05	0.15	0.40	0.30	0.10
分数	0~59	60~69	70~79	80~89	90~100											
比例数	0.05	0.15	0.40	0.30	0.10											
<pre>If (a<60) b=;tail;± else if (a<70) b=;pass;± else if (a<80) b=;general;± else if(a<90) b=;good;± else b=;excellent;±</pre>																
如图 (a) 所示																
																
以 5, 15, 40, 30, 10 为权构造哈夫曼树如图 (b) 所示，将判定框中的条件分开，可得到 (c)，从而实现判定过程的最优化。																
																
哈尔滨工业大学 计算机科学与技术学院 张岩																

数据结构与算法 第4章 树 Slide. 4 - 83				
4.5.3 哈夫曼 (Huffman) 树及其应用				
三、最优编码 (Huffman 编码)				
1. 问题的提出：				
编码 (如电报码) { 等长编码 不等长编码 } 特点：{ 编码长度 译码速度 传输速度 }				
前缀码 与 非前缀码				
例：CAST CATS SAT AT A TASA				
$D = \{A, C, S, T\}$ 按出现的频率 $w = \{2, 7, 4, 5\}$				
				
以 w_p 为外节点 构造哈夫曼树				
哈尔滨工业大学 计算机科学与技术学院 张岩				

数据结构与算法 第4章 树 Slide. 4 - 84				
4.5.3 哈夫曼 (Huffman) 树及其应用				
三、最优编码 (Huffman 编码)				
2. 编码和译码：				
▪ 编码 是指将文件 (字符集) 中的每个字符转换为一个唯一的二进制串。				
▪ 译码 (解码) 是指将二进制串转换为对应的字				
▪ 对于给定的字符集，可能存在多种编码方案，但应选择最优的。				
3. 编码的前缀性：				
▪ 前缀 对字符集进行编码时，如果任意一个字符的编码都不是其编码 它任何字符编码的前缀，则称这种编码具有前缀性或前缀编码。				
▪ 注意 ✓ 等长编码具有前缀性；				
✓ 变长编码可能使译码产生二义性，即不具有前缀性。 如，E(00), T(01), W(0001)，则译码时无法确定信息串是ET还是W。				
哈尔滨工业大学 计算机科学与技术学院 张岩				

数据结构与算法 第4章 树 Slide. 4 - 85

4.5.3 哈夫曼 (Huffman) 树及其应用

三、最优编码 (Huffman 编码)

4.字符集的平均编码长度：

- 设文件的字符集由 n 个不同字符构成 $C = \{c_1, c_2, \dots, c_n\}$ ，每个字符 c_j 的频度（出现比率）为 f_j ，码长为 l_j ，则 $\sum f_j \cdot l_j$ 表示编码后的文件总长。
- 通过对定义在相同字符集上的大量文件进行统计分析，得出每个字符 c_j 的概率（出现比率） p_j ，此时的 $\sum p_j \cdot l_j$ 表示平均码长。
- 把使得 $\sum f_j \cdot l_j$ 或 $\sum p_j \cdot l_j$ 最小的前缀编码称为最优的前缀码。

字符	a	b	c	d	e	f	平均
概率	0.45	0.13	0.12	0.16	0.09	0.05	码长
等长	000	001	010	011	100	101	3
变长	0	101	100	111	1101	1100	$= \lceil \log_2 C \rceil$

$= \sum p_j \cdot l_j$

- 最优的前缀码对文件的压缩效果也是最佳的。

哈尔滨工业大学 计算机科学与技术学院 张岩

数据结构与算法 第4章 树 Slide. 4 - 86

4.5.3 哈夫曼 (Huffman) 树及其应用

三、最优编码 (Huffman 编码)

5.Huffman 编码问题和编码算法：

- 对于给定的字符集以及每个字符出现的概率（使用频度），求字符集的最优的前缀性编码—Huffman 编码问题。
- 用 huffman 算法求字符集最优编码的算

- 使字符集中的每个字符对应一株只有叶结点的二元树，叶的权值为对应字符的使用频率；
- 利用 huffman 算法来构造一株 huffman 树；
- 对 huffman 树上的每个结点，左支附以 0，右支附以 1（或者相反），则从根到叶的路上的 0、1 序列就是相应字符的编码

哈尔滨工业大学 计算机科学与技术学院 张岩

数据结构与算法 第4章 树 Slide. 4 - 87

4.5.3 哈夫曼 (Huffman) 树及其应用

三、最优编码 (Huffman 编码)

5.Huffman 编码问题和编码算法：

- 举例

字符	a	b	c	d	e	f	平均
概率	0.45	0.13	0.12	0.16	0.09	0.05	码长
编码	0	101	100	111	1101	1100	2.24

ch	bits
0	a 0
1	b 101
2	c 100
3	d 111
4	e 1101
5	f 1100

编码表 H

哈尔滨工业大学 计算机科学与技术学院 张岩

数据结构与算法 第4章 树 Slide. 4 - 88

4.5.3 哈夫曼 (Huffman) 树及其应用

三、最优编码 (Huffman 编码)

6.Huffman 编码算法的实现：

- 存储结构

```
typedef struct{
    char ch; // 存储字符
    char bits[n+1]; // 字符编码位串
}CodeNode;
```

```
typedef CodeNode HuffmanCode[n];
HuffmanCode H;
```

ch	bits
0	a 0
1	b 101
2	c 100
3	d 111
4	e 1101
5	f 1100

编码表 H

哈尔滨工业大学 计算机科学与技术学院 张岩

哈工大计算机考研全套视频和资料，真题、考点、典型题、命题规律独家视频讲解！
详见：网学天地（www.e-studysky.com）；咨询QQ：2696670126

数据结构与算法	第4章 树	Slide. 4 - 89
■ 编码算法		
<pre>void CharSetHuffmanEncoding(HuffmanT T, HuffmanCode H) { /*根据Huffman树T求Huffman编码表H*/ int c, p, i; /* c 和p 分别指示T 中孩子和双亲的位置 */ char cd[n+1]; /* 临时存放编码 */ int start; /* 指示编码在cd 中的位置 */ cd[n]='0'; /* 编码结束符 */ for(i=0; i<n; i++){ /* 依次求叶子T[i]的编码 */ H[i].ch=getchar(); /* 读入叶子T[i]对应的字符 */ start=n; /* 编码起始位置的初值 */ c=i; /* 从叶子T[i]开始上溯 */ while((p=T[c.parent])>=0){ /* 直到上溯到T[c]是树根位置 */ cd[-start]=(T[p].lchild==c)? '0' : '1'; /* 若T[c]是T[p]的左孩子，则生成代码0，否则生成代码1*/ c=p; /* 继续上溯 */ } strcpy(H[i].bits,&cd[start]); /* 复制编码为串于编码表H*/ } }</pre>		

哈尔滨工业大学 计算机科学与技术学院 张岩

数据结构与算法	第4章 树	Slide. 4 - 90
4.5.3 哈夫曼（Huffman）树及其应用		
<p>三、最优编码（Huffman编码）</p> <p>7. 利用 Huffman 编码对数据文件编码和译码：</p> <ul style="list-style-type: none"> 编码 依次读入文件的字符 c，在 huffman 编码表 H 中找到此字符，若 H[i].ch == c，则将 c 转换为 H[i].bits 中的编码串。 译码 依次读入文件的二进制码。在 huffman 树中从根结点 T[m-1] 出发，若读入 0，则走左支，否则，走右支，一旦到达某叶结点 T[i] 时便译出相应的字符 H[i].ch。然后重新从根出发继续译码，直到文件结束。 <p>Huffman 编码一定具有前缀性；</p> <ul style="list-style-type: none"> Huffman 编码是最小冗余码； Huffman 编码方法，可以使出现概率大的字符对应的码长较短； Huffman 编码不唯一，可以用于加密； Huffman 编码译码简单唯一，没有二义性。 		

哈尔滨工业大学 计算机科学与技术学院 张岩

数据结构与算法	第4章 树	Slide. 4 - 91
4.5.4 树的应用—表达式求值		
<p>参考 4.4</p>		

哈尔滨工业大学 计算机科学与技术学院 张岩

数据结构与算法	第4章 树	Slide. 4 - 92
<p>树 ◦ 树的 ADT { 逻辑结构 存储结构 }</p> <p>◦ 二元树 { 逻辑结构 存储结构 } → 顺序存储 ◦ 基本性质 ◦ 遍历二元树 { 先根顺序 中根顺序 后根顺序 } → 递归 ◦ 线索二元树 → 层序遍历</p> <p>◦ 树的存储结构 → 双亲表示法(数组) 孩子表示法(邻接表) 左右链表示(二元树)</p> <p>◦ 树的应用 { 集合表示 哈夫曼树 } → 优化判定过程 哈夫曼编码 表达式求值</p>		
哈尔滨工业大学 计算机科学与技术学院 张岩		