第08讲 线索化二叉树和哈夫曼树

上课老师：六星教育-新雨老师

7.6 线索化二叉树

7.7 树和森林

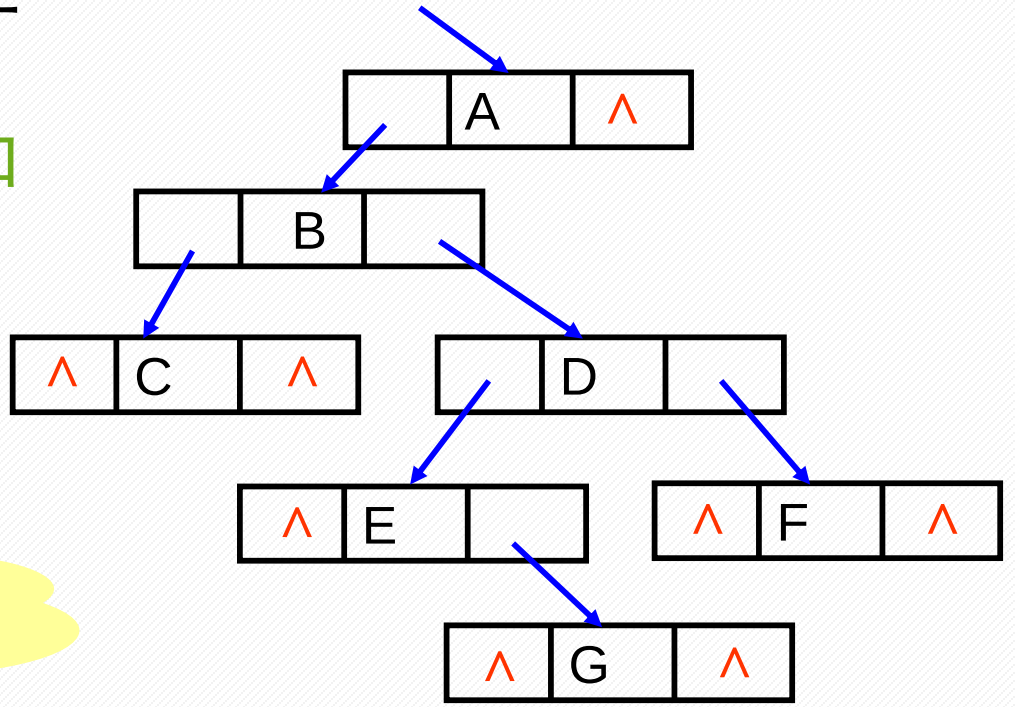
7.8 哈夫曼树及其应用

思考分析

在n个结点的二叉链表中，有n+1个空指针域

二叉链表空间效率这么低，能否利用这些空闲区存放有用的信息或线索？

——可以用它来存放当前结点的直接前驱和后继等线索，以加快查找速度。



线索化二叉树

7.6 线索化二叉树

普通二叉树只能找到结点的左右孩子信息，而该结点的直接前驱和直接后继只能在遍历过程中获得

若将遍历后对应的有关前驱和后继预存起来，则从第一个结点开始就能很快“顺藤摸瓜”而遍历整个树

可能是根、或最左（右）叶子

线索化二叉树

如何保存这类信息?

两种解决方法

- 增加两个域：fwd和bwd；
- 利用空链域（n+1个空链域）

线索化二叉树

1）若结点有左子树，则lchild指向其左孩子；
　　否则， lchild指向其直接前驱(即线索)；

2）若结点有右子树，则rchild指向其右孩子；
　　否则， rchild指向其直接后继(即线索) 。

为了避免混淆，增加两个标志域

lchild	LTag	data	RTag	rchild

线索化二叉树

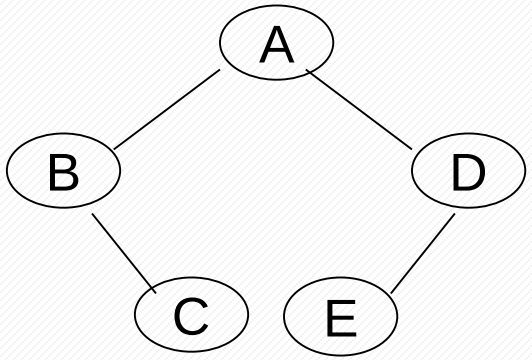
lchild	LTag	data	RTag	rchild

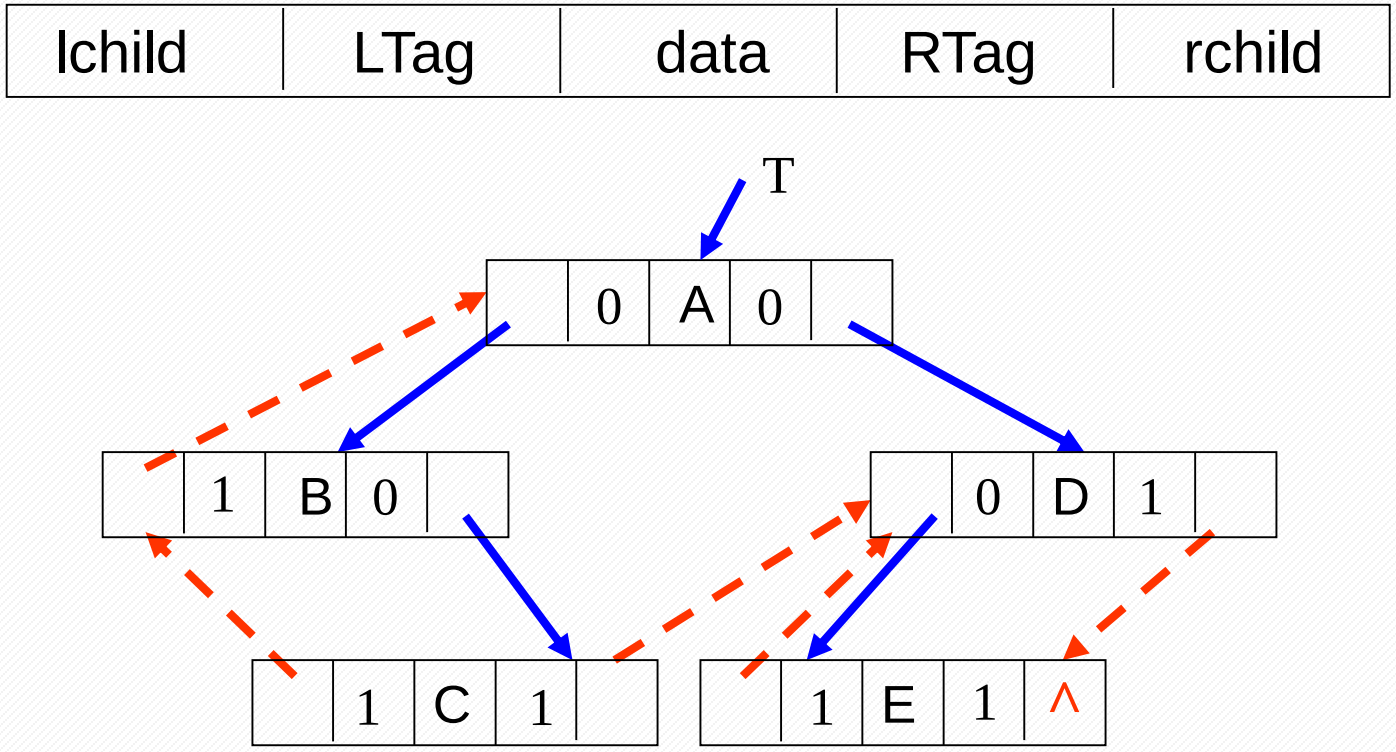
Ltag: 若 LTag=0, lchild域指向左孩子；
若 LTag=1, lchild域指向其前驱。

RTag: 若 RTag=0, rchild域指向右孩子；
若 RTag=1, rchild域指向其后继。

线索化二叉树

LTag=0, lchild域指向左孩子
LTag=1, lchild域指向其前驱
RTag=0, rchild域指向右孩子
RTag=1, rchild域指向其后继

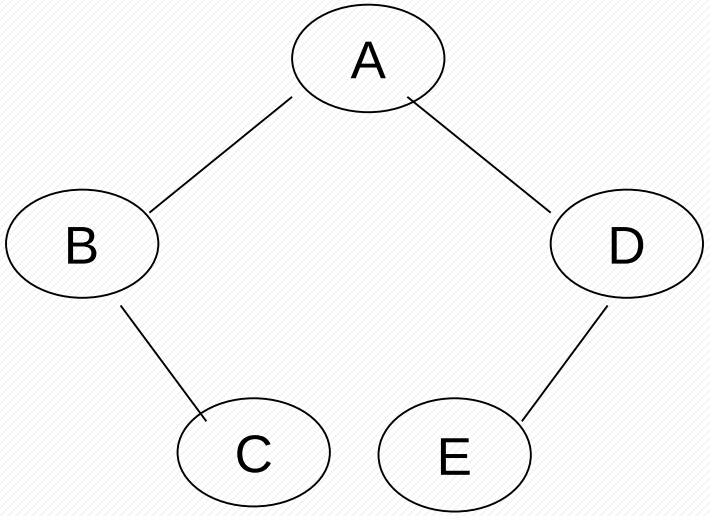


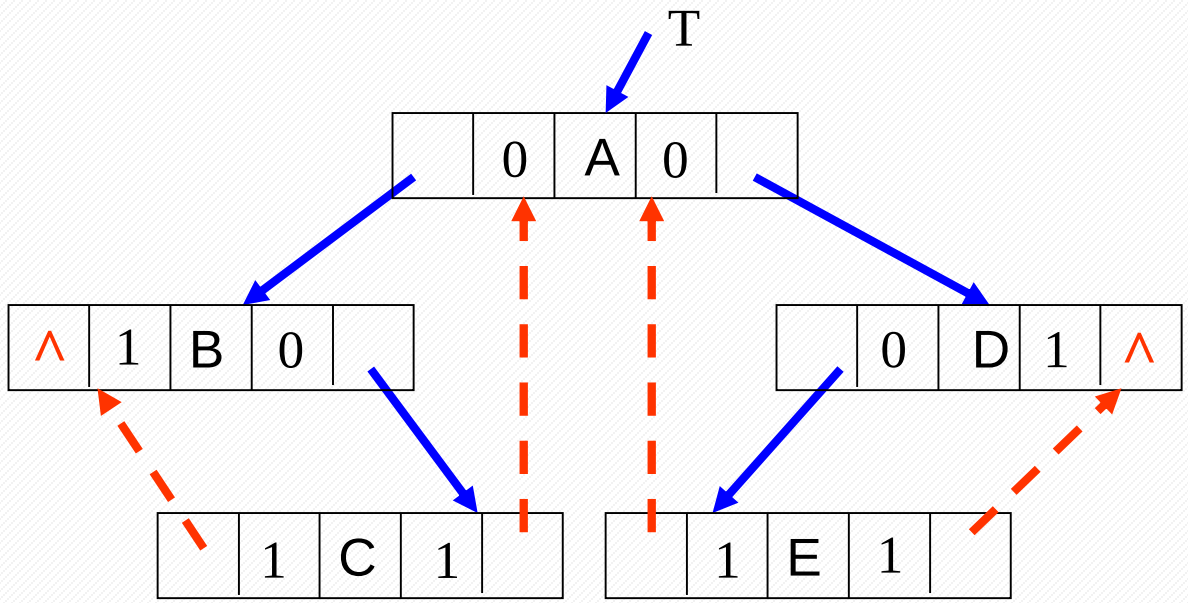


先序序列：ABCDE

线索化二叉树

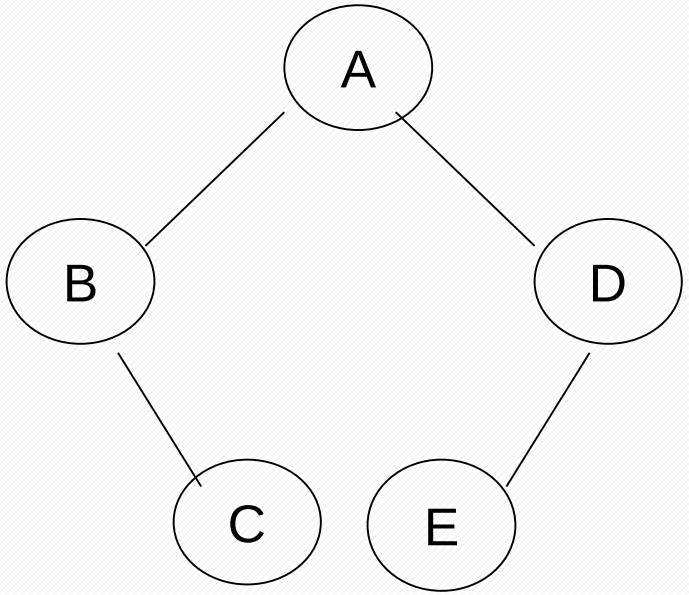
lchild	LTag	data	RTag	rchild



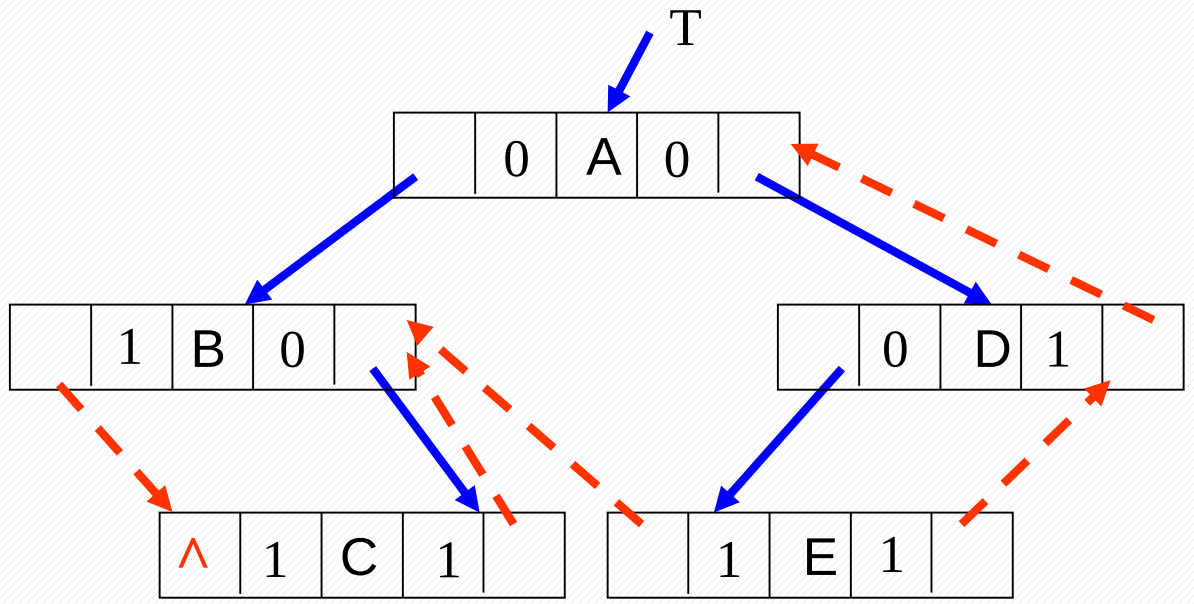


中序序列：BCAED

线索化二叉树



lchild	LTag	data	RTag	rchild



后序序列：CBEDA

线索化二叉树的几个术语

线索：指向结点前驱和后继的指针

线索链表：加上线索二叉链表

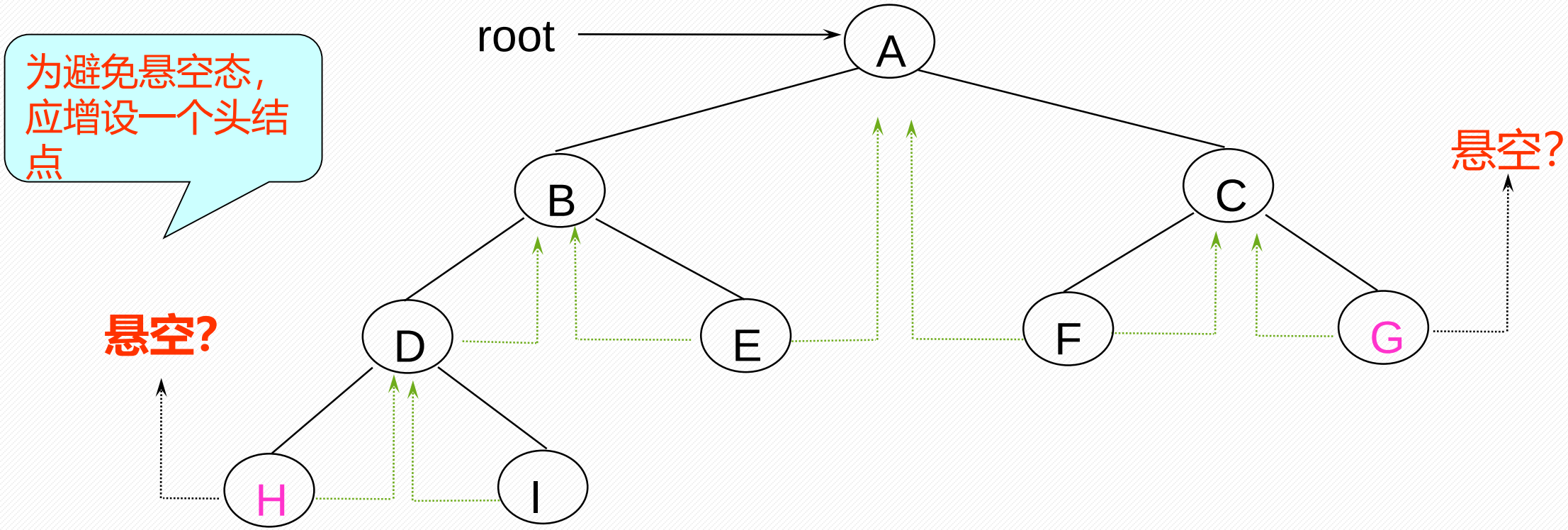
线索二叉树：加上线索的二叉树（图形式样）

线索化：对二叉树以某种次序遍历使其变为线索二叉树的过程

综合练习分析

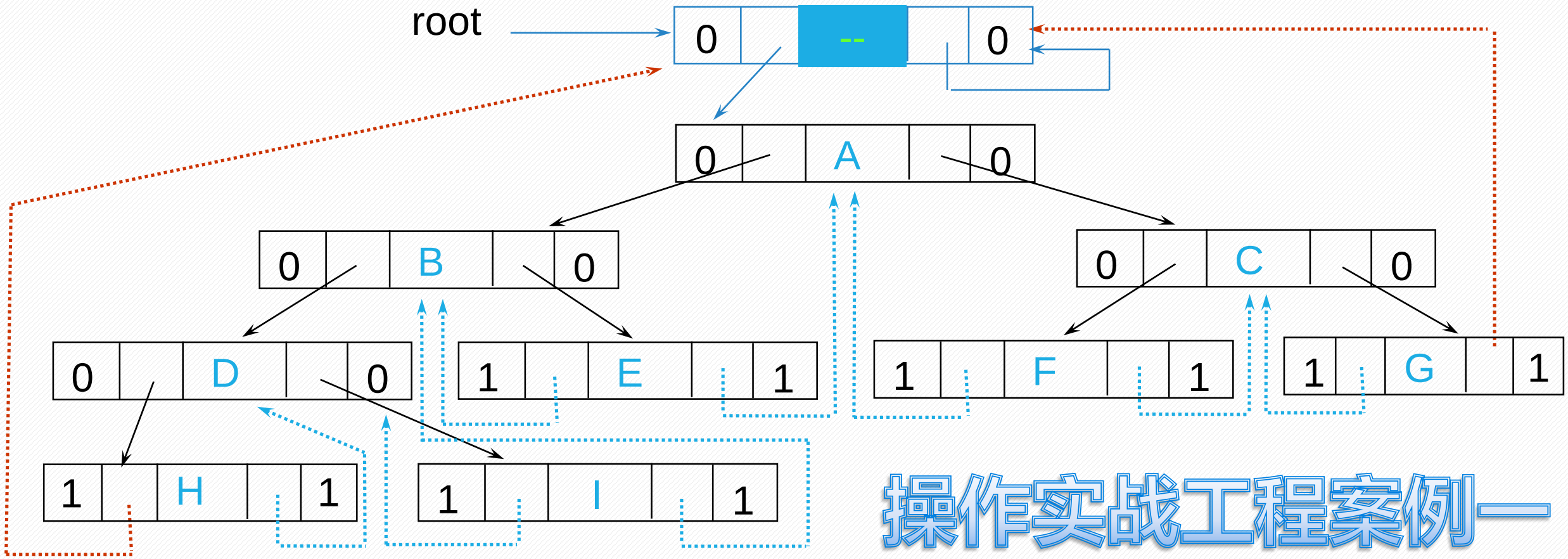
画出以下二叉树对应的中序线索二叉树。

该二叉树中序遍历结果为：H, D, I, B, E, A, F, C, G



对应中序线索二叉树存储结构如图所示：

注：此图中序遍历结果为：H, D, I, B, E, A, F, C, G



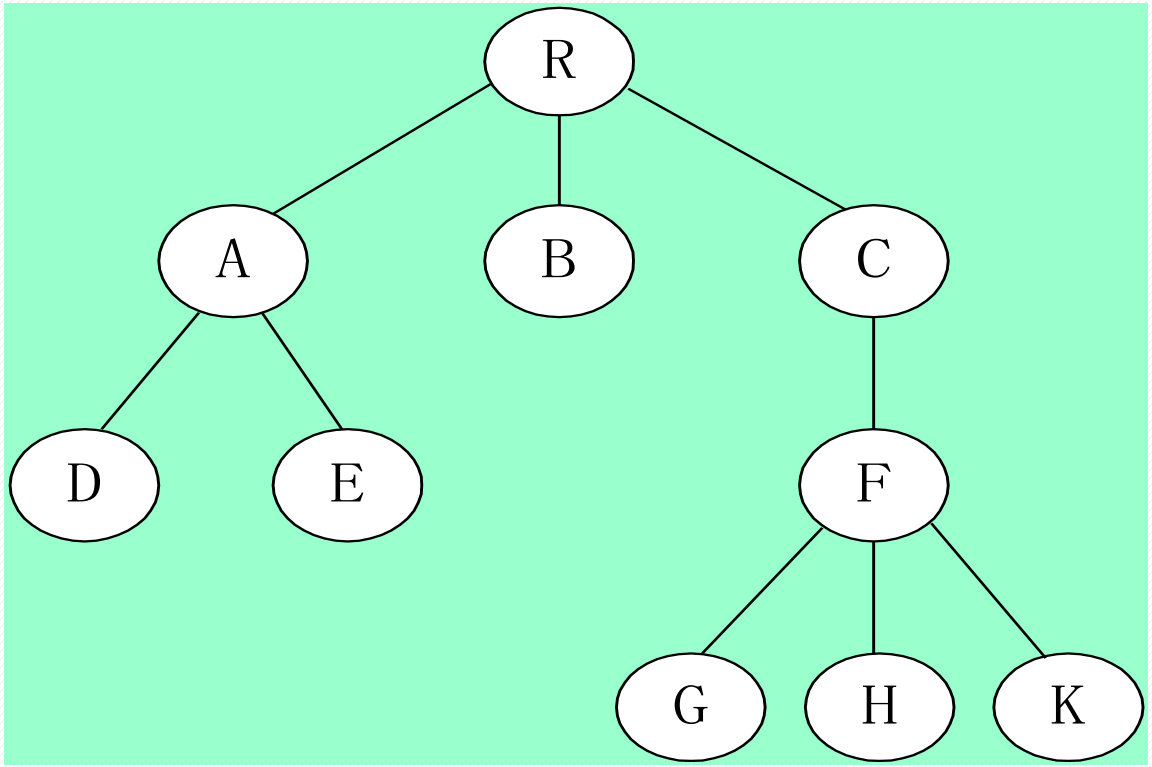
操作实战工程案例一

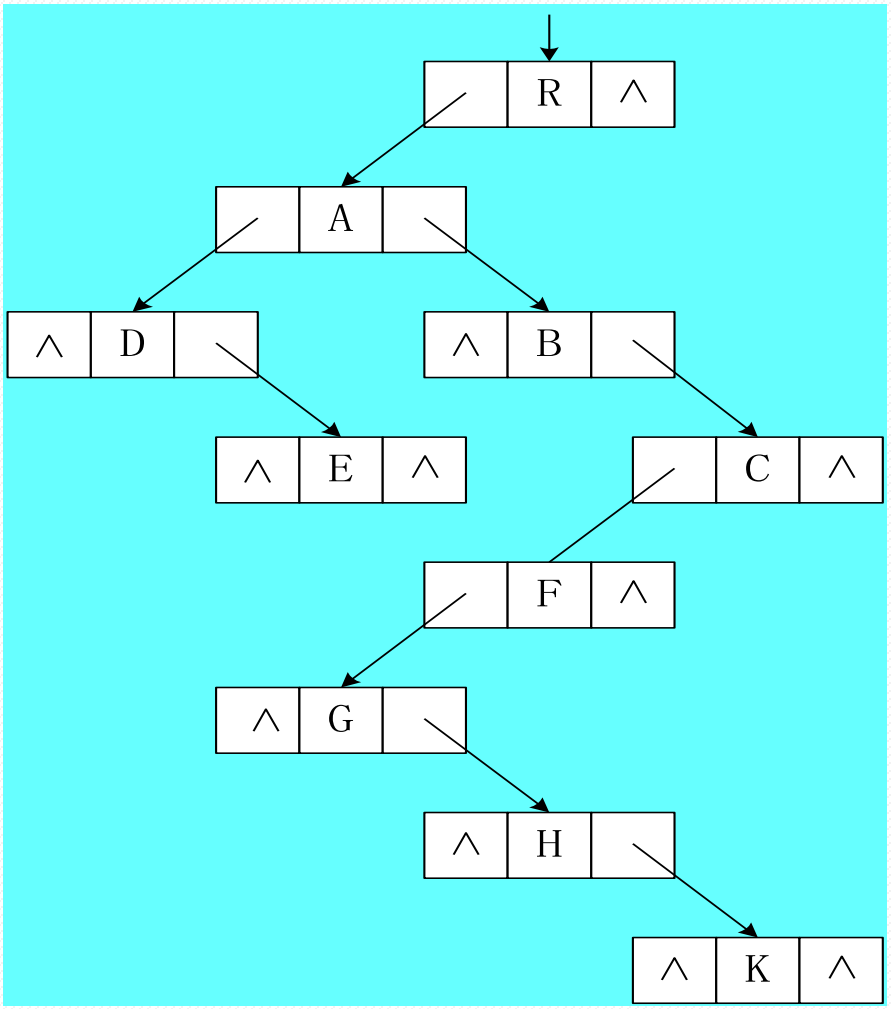
7.7 树和森林

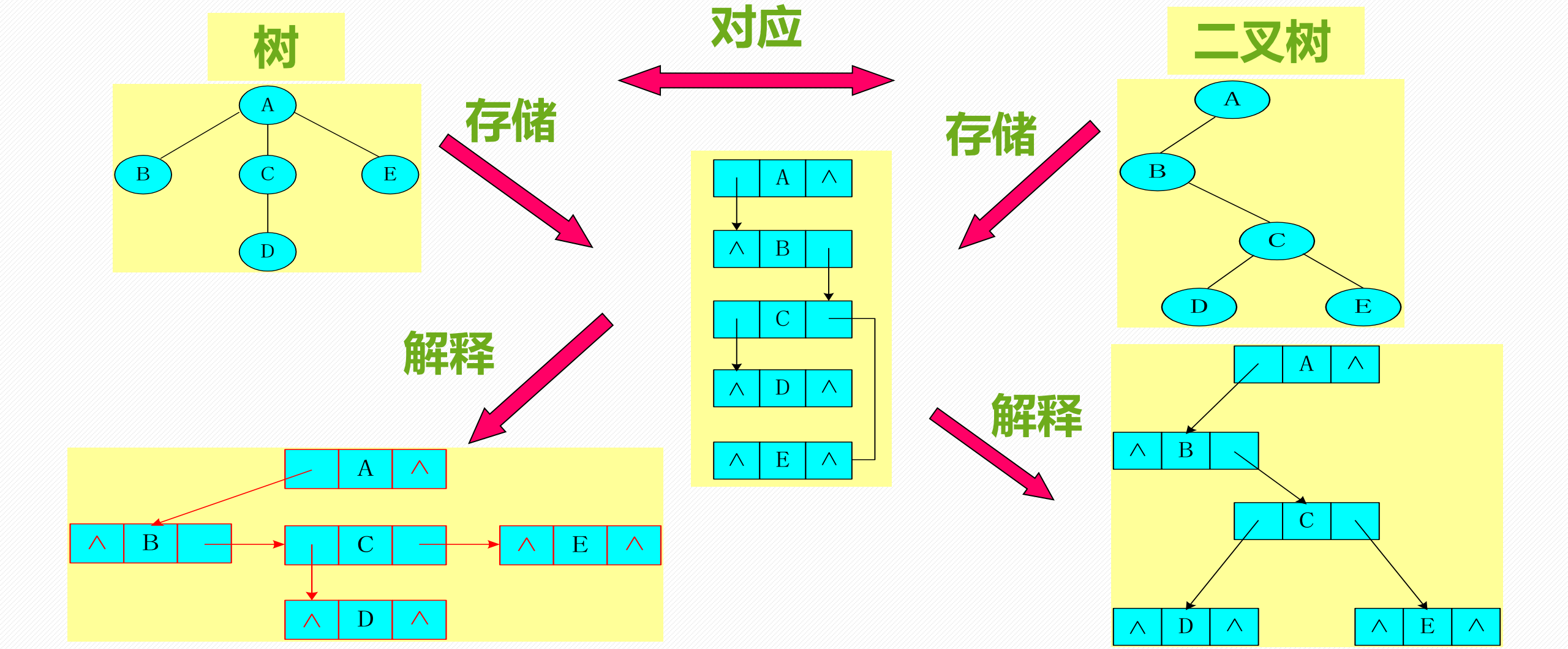
树的存储结构-二叉链表表示法

```
typedef struct CSNode
{
        ElemType   data;
        struct CSNode  *firstchild, *nextsibling;
} CSNode,*CSTree;
```

树的存储结构-二叉链表表示法






树
对应
二叉树
存储
存储
解释
解释
A
B
C
D
E
^

1、树转换为二叉树

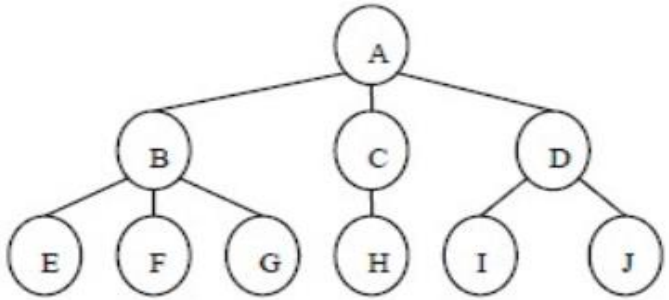
由于二叉树是有序的，为了避免混淆，对于无序树，我们约定树中的每个结点的孩子结点按从左到右的顺序进行编号。

将树转换成二叉树的步骤是：

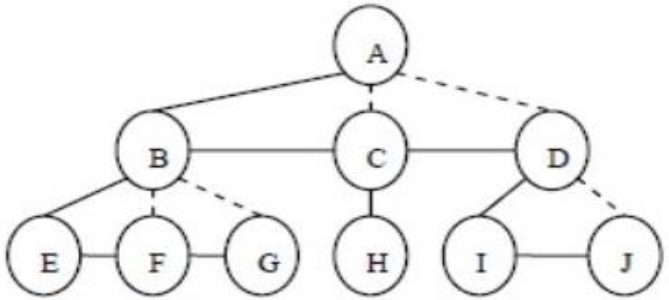
（1）***加线***。就是在所有兄弟结点之间加一条连线；

（2）***抹线***。就是对树中的每个结点，只保留他与第一个孩子结点之间的连线，删除它与其它孩子结点之间的连线；

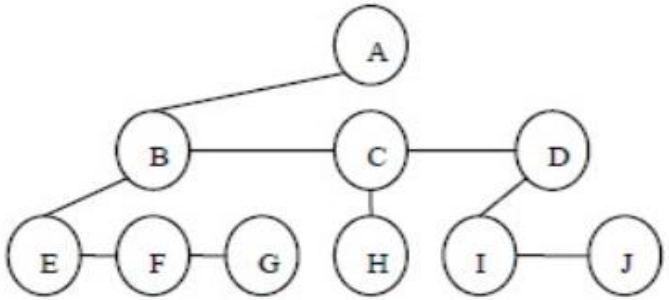
（3）***旋转***。就是以树的根结点为轴心，将整棵树顺时针旋转一定角度，使之结构层次分明。



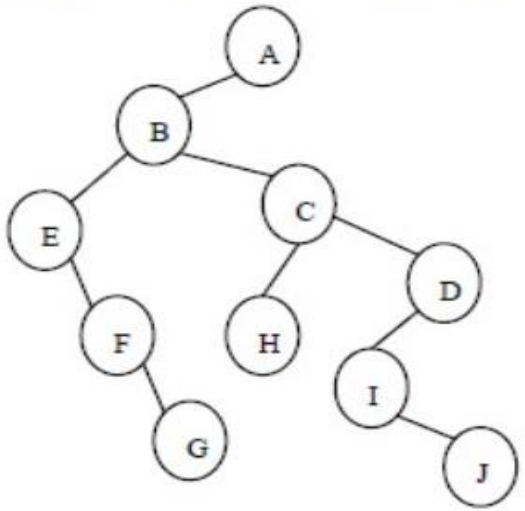
(a) 树



(b) 加线（虚线表示要删除的线）



(c) 抹线



(d) 旋转

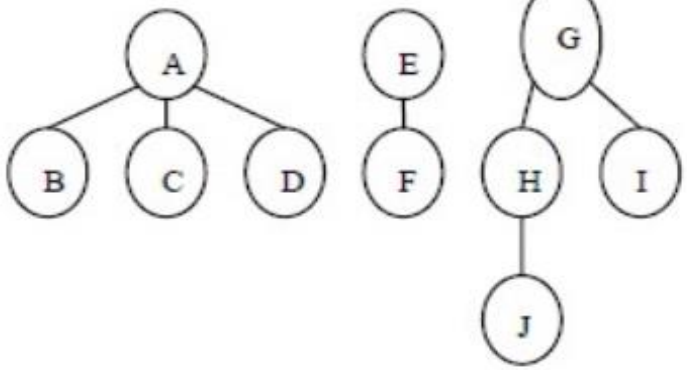
2、森林转换为二叉树

森林是由若干棵树组成，可以将森林中的每棵树的根结点看作是兄弟，由于每棵树都可以转换为二叉树，所以森林也可以转换为二叉树。

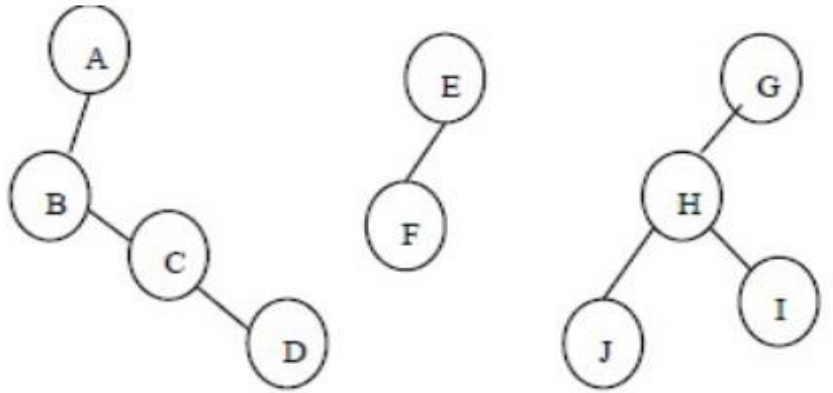
将森林转换为二叉树的步骤是：

（1）先把每棵树转换为二叉树；

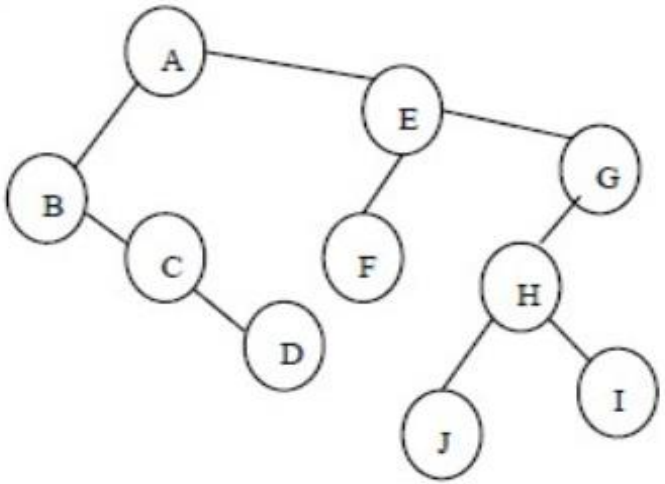
（2）第一棵二叉树不动，从第二棵二叉树开始，依次把后一棵二叉树的根结点作为前一棵二叉树的**根结点的右孩子结点**，用线连接起来。当所有的二叉树连接起来后得到的二叉树就是由森林转换得到的二叉树。



(a) 森林



(b) 森林中每棵树转换的二叉树



(c) 森林转化得到的二叉树

7.8 哈夫曼树及其应用

- 游戏中主角的生命值d，有这样的条件判定：当怪物碰到主角后，怪物的反应遵从下规则：

条件：	d<100	100<=d<200	200<=d<300	300<=d<500	d>500
反应：	嘲笑，单挑	单挑	嗜血魔法	呼唤同伴	逃跑

条件：	d<100	100<=d<200	200<=d<300	300<=d<500	d>500
反应：	嘲笑，单挑	单挑	嗜血魔法	呼唤同伴	逃跑


```
if(d<100) state=嘲笑，单挑;
    else if(d<200) state=单挑;
        else if(d<300) state=嗜血魔法;
            else if(d<500) state=呼唤同伴;
                else state=逃跑;
```

哈夫曼树应用实例-哈夫曼编码

在远程通讯中，要将待传字符转换成二进制的字符串，怎样编码才能使它们组成的报文在网络中传得最快?

A	00
B	01
C	10
D	11

ABACCDA

A	0
B	00
C	1
D	01

00 01 00 10 10 11 00

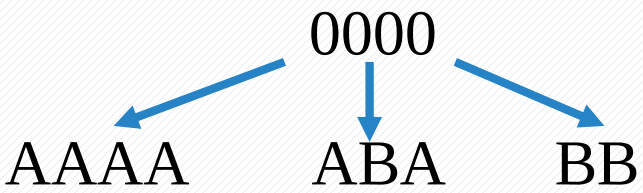
0 00 0 1 1 01 0

出现次数较多的字符采用尽可能短的编码

哈夫曼树应用实例-哈夫曼编码

ABACCDA

A	0
B	00
C	1
D	01



重码

0 00 0 1 1 01 0

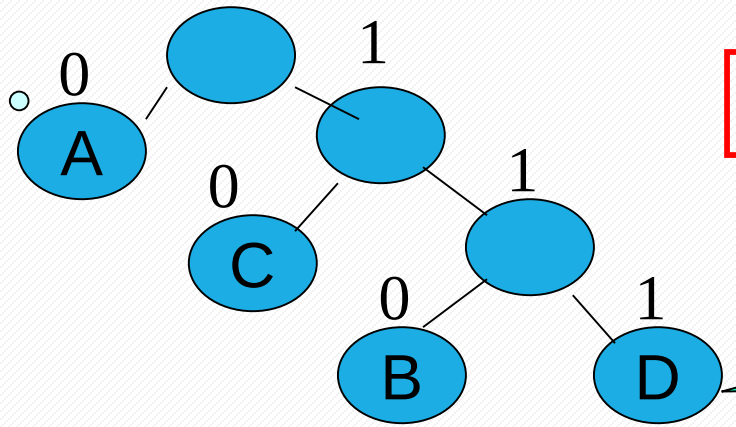
关键：要设计长度不等的编码，则必须使任一字符的编码都不是另一个字符的编码的前缀 - 前缀编码

采用二叉树设计前缀编码

ABACCDA

A—0
B—110
C—10
D—111

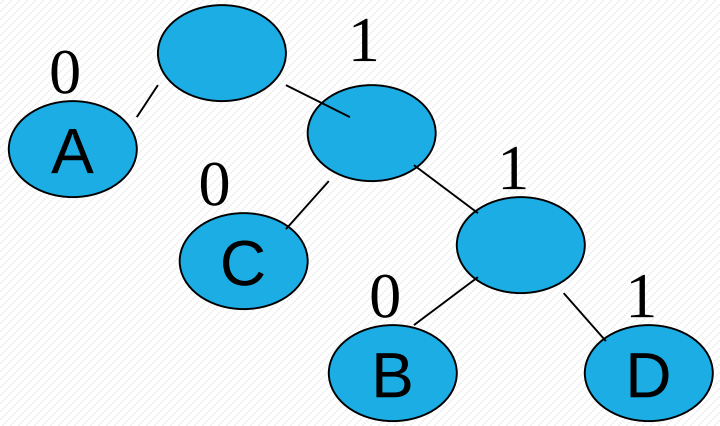
0 110 0 10 10 111 0



左分支用“0”
右分支用“1”

哈夫曼编码的译码过程

分解接收字符串：遇“0”向左，遇“1”向右；一旦到达叶子结点，则译出一个字符，反复由根出发，直到译码完成。



0 110 0 10 10 111 0

ABACCDA

特点：每一码都不是另一码的前缀，绝不会错译! 称为前缀码

哈夫曼树的构造

路径： 由一结点到另一结点间的分支所构成

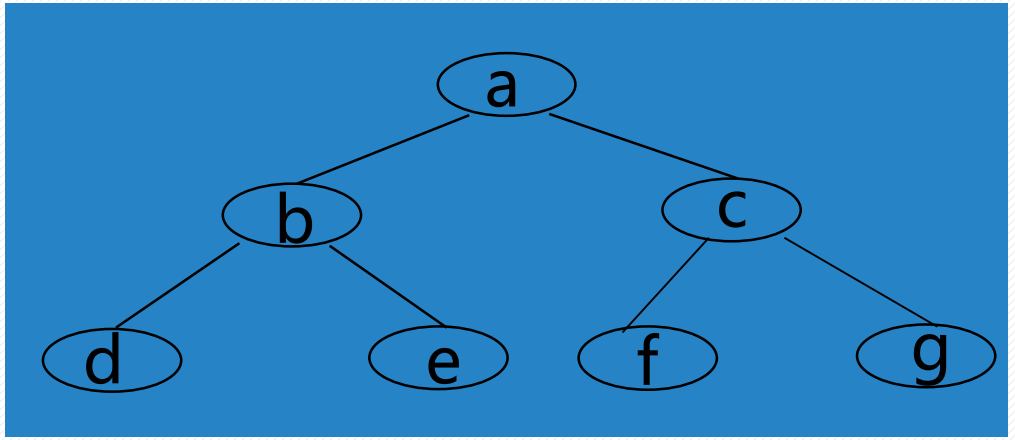
路径长度： 路径上的分支数目 a→e的路径长度 = 2

带权路径长度： 结点到根的路径长度与结点上权的乘积

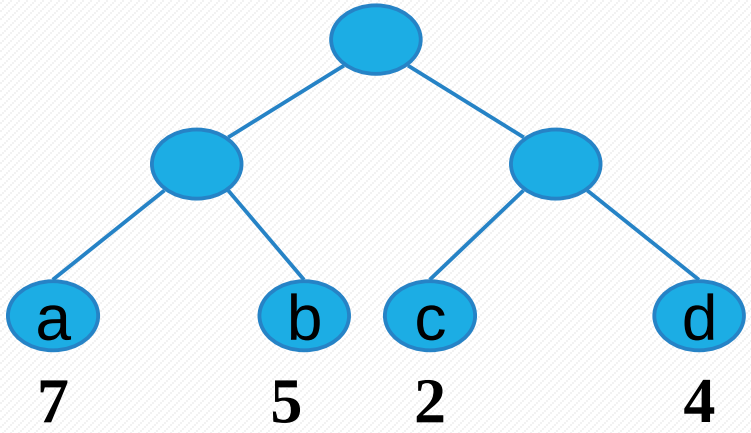
树的带权路径长度：树中所有叶子结点的带权路径长度之和

$$WPL = \sum_{k=1}^{n} w_k l_k$$

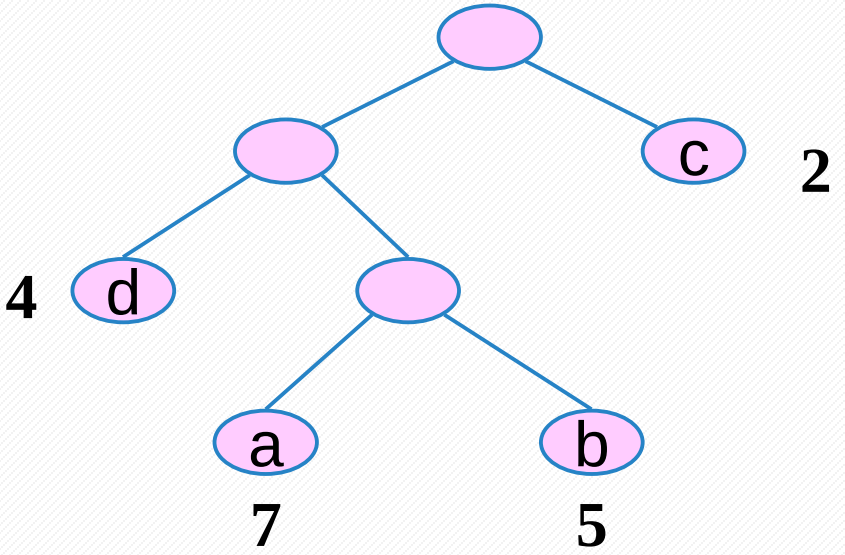
哈 夫 曼 树： 带权路径长度最小的树



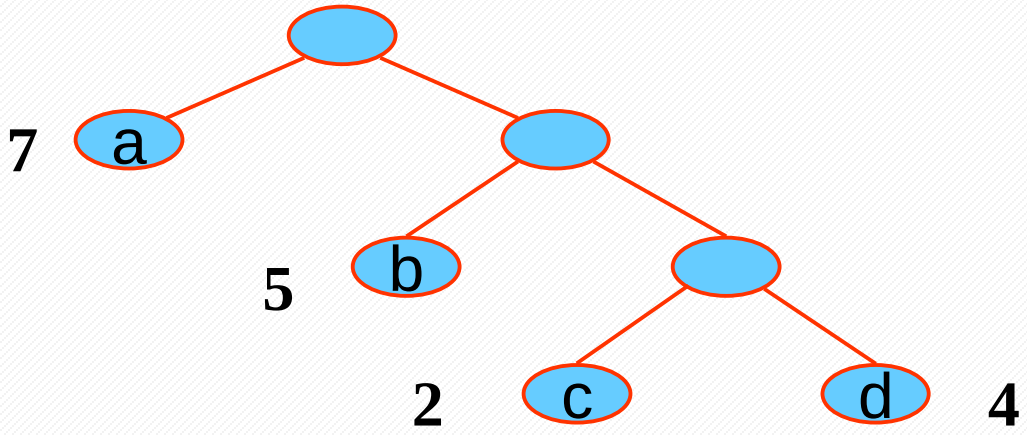
构造有4个叶子结点的二叉树



WPL=7*2+5*2+2*2+4*2=36



WPL=7*3+5*3+2*1+4*2=46

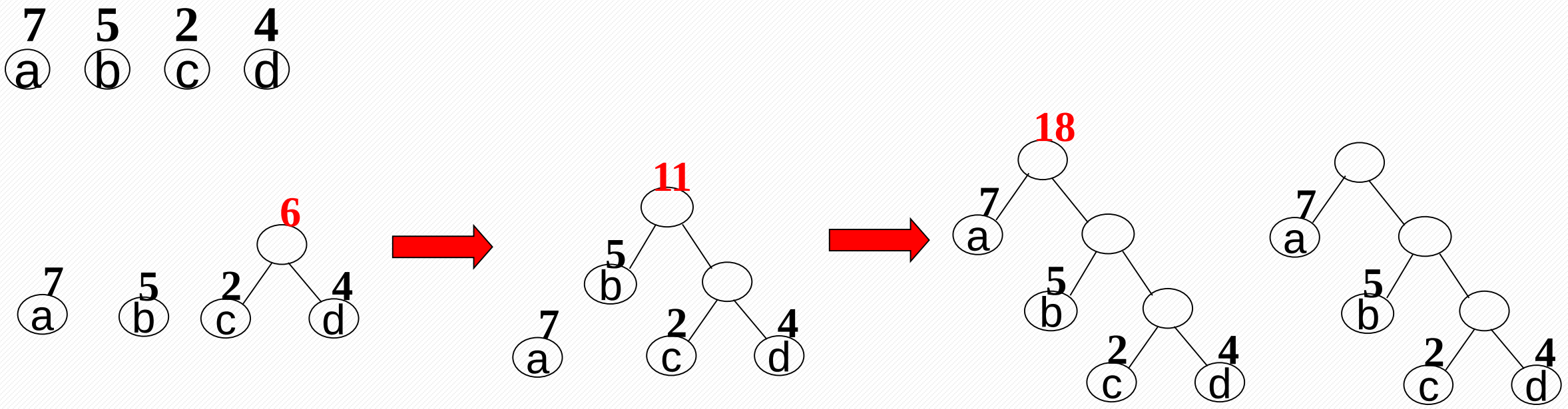


WPL=7*1+5*2+2*3+4*3=35

哈夫曼树的构造过程

基本思想：使权大的结点靠近根

操作要点：对权值的合并、删除与替换，总是合并当前值最小的两个



哈夫曼树的构造过程

- ✓根据给定的n个权值{w_1,w_2,……w_n}，构造n棵只有根结点的二叉树。
- ✓在森林中选取两棵根结点权值最小的树作左右子树，构造一棵新的二叉树，置新二叉树根结点权值为其左右子树根结点权值之和。
- ✓在森林中删除这两棵树，同时将新得到的二叉树加入森林中。
- ✓重复上述两步，直到只含一棵树为止，这棵树即哈夫曼树。

哈夫曼树构造算法的实现

1、初始化HT[1..2n-1]：lch=rch=parent=0

2、输入初始n个叶子结点：置HT[1..n]的weight值

3、进行以下n-1次合并，依次产生HT[i]，i=n+1..2n-1：

3.1 在HT[1..i-1]中选两个未被选过的weight最小的两个结点HT[s1]和HT[s2] (从parent = 0 的结点中选)

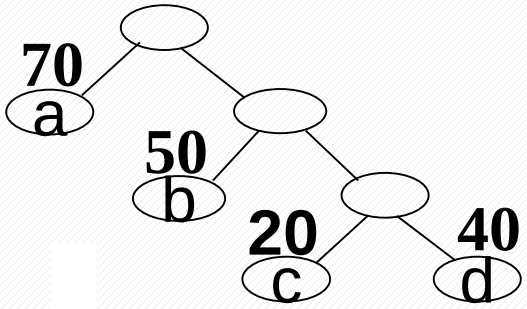
3.2 修改HT[s1]和HT[s2]的parent值： parent=i

3.3 置HT[i]：weight=HT[s1].weight + HT[s2].weight ,

lch=s1，rch=s2

案例:设n=4，w = {70,50,20,40}

试设计 huffman code (m=2*4-1=7)

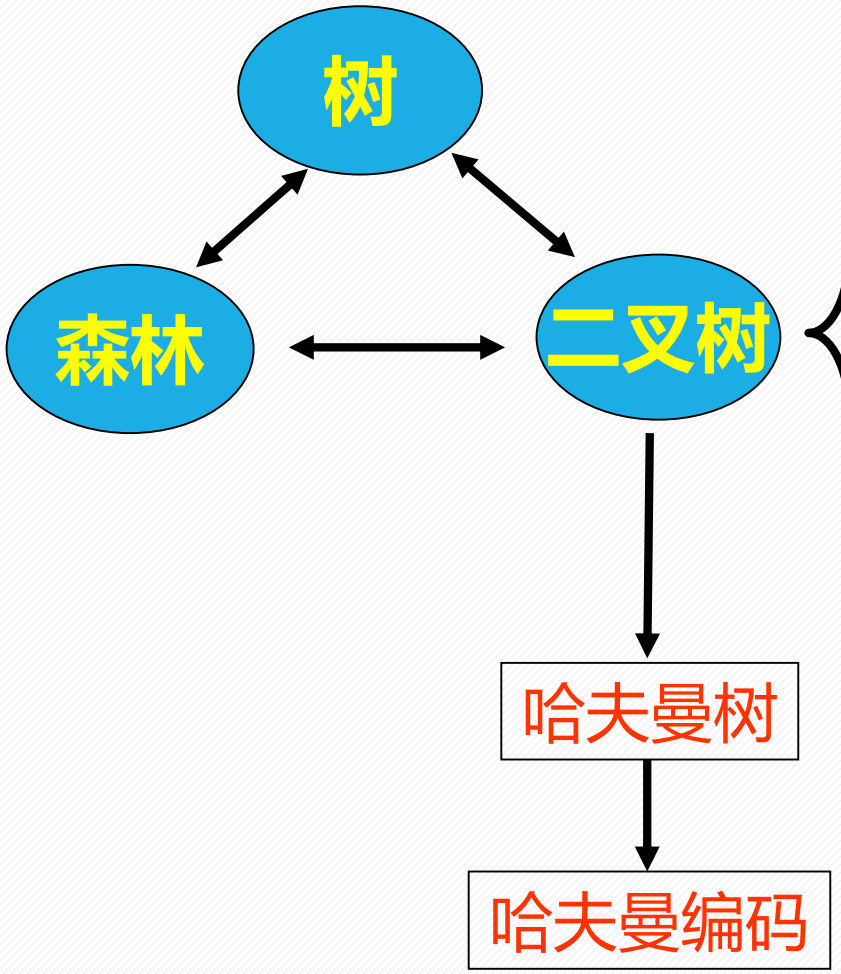


	weight	parent	lch	rch
1	70	0	0	0
2	50	0	0	0
3	20	0	0	0
4	40	0	0	0
5				
6				
7				

	weight	parent	lch	rch
1	70	7	0	0
2	50	6	0	0
3	20	5	0	0
4	**40**	**5**	**0**	**0**
5	**60**	**6**	**3**	**4**
6	**110**	**7**	**2**	**5**
7	**180**	**0**	**1**	**6**

哈夫曼编码的几点结论

- 哈夫曼编码是不等长编码
- 哈夫曼编码是前缀编码，即任一字符的编码都不是另一字符编码的前缀
- 哈夫曼编码树中没有度为1的结点。若叶子结点的个数为n，则哈夫曼编码树的结点总数为 2n-1
- 发送过程：根据由哈夫曼树得到的编码表送出字符数据
- 接收过程：按左0、右1的规定，从根结点走到一个叶结点，完成一个字符的译码。反复此过程，直到接收数据结束



1、定义和性质

2、存储结构
- 顺序结构
- 链式结构
  - 二叉链表
  - 三叉链表

3、遍历
- 先序遍历
- 中序遍历
- 后序遍历

4、线索化：线索树

操作实战工程案例二

CimFAX
传真服务器常用
问答

将来的你

冲刺 加油!

让我们全力以赴

一定会感激现在拼命的自己