1. 树与二叉树

6.1 树的基本概念

1. 树的定义
2. D：树是**相同类型**的n个元素的有限集合，是有层次的非线性结构

——>>Compare：广义表与树的区别

对于非首非尾节点，广义表仍然只有一个前驱与一个后继，而树具有一个前驱与多个后继

1. R：树的数据关系
2. 若D为空集，则此树为空树
3. 树中存在唯一的根节点root
4. 当n>0时，每个树的子集也能构成一棵子树
5. 节点定义：元素域+左孩子+右兄弟
6. 树的基本术语
7. 节点的度：每个节点拥有的子树数目，也即分支数目
8. 树的度：树中所有节点的度的最大值
9. 叶子节点：度为0的节点
10. 树的深度：叶子节点所在的最大层次
11. 森林：m（m>=0）棵互不相交的树的集合，使用根节点与森林名的二元集合表示

6.2二叉树

1. 二叉树的定义
2. 二叉树每个节点之多只有两棵子树（但是不一定只有两层）
3. 只有五种形态：空、根、仅左、仅右、左右
4. 满二叉树：只含有度为0或2的节点
5. 完全二叉树：最下面一层的节点从左到右排列，上面的各层全满

//完全二叉树是可以使用顺序表进行存储的，每个序号的节点对应顺序下标

1. 二叉树的性质
2. 二叉树的第i层上之多有2i-1个节点
3. 深度为k的二叉树上至多有2k-1个节点
4. 对于任何二叉树，若有n0个子节点，n2个度为2的节点，则必有n0=n2+1

//n个节点的二叉树有n-1条边

1. 具有n个节点的完全二叉树的深度是
2. 二叉树的存储
3. 二叉链表存储

Left-child data right-child

1. 三叉链表存储

Left-child data parent right-child

6.3 二叉树的遍历

一、遍历的定义

0、遍历操作：前序、中序、后序的区别在于V的位置

访问根节点V

访问左子树L

访问右子树R

使用递归方法描述，若二叉树为空则空操作

也可以使用栈来非递归实现

1. 前序遍历VLR：压栈与深度优先搜索DFS（回溯）
2. 中序遍历LVR
3. 后序遍历LRV
4. 层次遍历（从上到下，从左到右）：广度优先搜索BFS
5. 遍历的应用
6. 计算二叉树的节点个数：如果子树指针非空则递归求其左右子树，否则返回0
7. 由给定表达式建立相应二叉树：
   1. 若表达式中仅有一个数或变量，则树中只有一个根节点

//（操作数也可以是表达式）

* 1. 若是二元运算符x operator y 则左子树表示x，右子树表示y，根节点表示op

1. 三种遍历中由二推三（但有一种不行）

由二叉树的前序序列和中序序列可以唯一地确定一棵二叉树

　　由二叉树的后序序列和中序序列可以唯一地确定一棵二叉树

由二叉树的**前序**序列和**后序**序列**不可以**唯一地确定一棵二叉树

* 1. 重点在于先确定根节点
  2. 然后通过根节点位置来确定哪些在左子树哪些在右子树
  3. 在子树中重复上述操作，先确定根节点然后不断分割子树
  4. 最终子树只剩下两三个元素则易于确定

6.4线索二叉树

1. 线索二叉树的定义
2. 问题提出：遍历二叉树的结果是节点的一个线性序列，能否不找完整个O(n)遍历而找到遍历结果中某个节点的前驱与后继O(1)
3. 线索：指向该线性序列中的“前驱”和“后继”的指针称为线索
4. 线索二叉树：利用原节点中的“空指针”来存储“线索”，包含线索的存储结构称为线索链表，使用这样的结构来建立的树称为线索二叉树

**Attention：并不是所有的节点都含有空指针——>问题所在**

1. 线索二叉树的构建与遍历
2. 寻找当前节点在中序遍历下的后继：
   1. 若无右子树，则为其后继线索所指节点
   2. 否则为对其右子树做中序遍历找到的第一个节点
3. 寻找当前节点在中序遍历下的前驱：
   1. 若无左子树，则为其后继线索所指节点
   2. 否则为对其左子树做中序遍历找到的最后一个节点
4. 前序的前驱/后序的后继较为复杂

6.5 树与森林

一、树与森林的表示

1、子树链表法

2、孩子双亲法

3、孩子兄弟法

1. 森林与二叉树的转换

首先将森林中的每一棵树转化为二叉树,其基本方法是对树中的每一个结点,都转化为一个二叉树结点,各树结点的第一个孩子作为它在二叉树中的左孩子,而将树结点的右兄弟转化为在二叉树中的右孩子,其余结点依此类推。将森林中的各棵树都转化为对应的二叉树表示后,取第一棵树的根作为最终二叉树的根,第二棵树对应的二叉树作为最终二叉树的右子树,对森林中剩下的树实行同样的操作,即可得到对应的二叉树。

1. 树与森林的遍历
2. 森林的先序遍历（二叉树的先序遍历）
   1. 先访问第一棵树的根节点
   2. 先根遍历第一棵树的子树森林
   3. 先根遍历其他树组成的森林
3. 森林的后根遍历（二叉树的中序遍历）
   1. 中序遍历第一棵树的子树森林
   2. 访问第一棵树的根节点
   3. 中序遍历其他树构成的森林
4. 树的计数

具有n个节点的不同形态的树有多少棵？

结论：含有n个节点的不相似的树的个数是

6.6 哈夫曼编码与哈夫曼树

一、哈夫曼编码

使用不等长编码对原文进行编码，出现次数更高的词语使用短编码

1. 哈夫曼树
2. 在所有含n个叶子节点并带相同权值的m叉树中，比存在一棵带权路径最小的树，称为“最优树”或“哈夫曼树”
3. 在哈夫曼树中，权值最大的节点离根节点最近
4. 前缀码：在一个编码系统中，任何一个编码都不是其他任何编码的前缀，则称此系统是前缀码。不定长编码一定要是前缀码，否则会产生歧义
5. 哈夫曼树的构建与解码

(1)根据给定的n个权值{w1,w2,…,wn},构造由n棵二叉树构成的森林F={T1,T2,…,T.},其中每棵二叉树T1分别都是只含有一个带权值为w;的根结点,其左、右子树为空(i=1,2,…,n);

(2)在森林F中选取其根结点的权值为最小的两棵二叉树(若这样的二叉树不止两棵时,则任选其中两棵),分别作为左、右子树构造一棵新的二叉树,并置这棵新的二叉树根结点的权值为其左、右子树根结点的权值之和;

(3)从森林F中删去这两棵二叉树,同时将刚生成的新二叉树加入到森林F中;

(4)重复(2)和(3)两步骤,直至森林F中只含一棵二叉树为止。

最后得到的那棵二叉树就是哈夫曼树。

——>哈夫曼树的解码：

从根节点走到叶节点，左分支赋予0，右分支赋予1，直到叶节点，可获得叶节点编码

**讲人话：**

