树：n（n>0）个结点的有限集；

对非空树（n>0）：①有且仅有一个特定的称为根（root)的结点；

②其余结点可以分为m（m>0）个互不相交的有限集，其中每个集合本身又是一棵树，称为原来的树的子树（SubTree）；

1.子树是不相交的；2.除了根结点外，每个结点有且仅有一个父结点；3.一颗N个结点的树有N-1条边；

结点的度（Degree）：结点的子树个数；

树的度：树的所有结点中最大的度数；

叶结点（叶子Leaf或终端结点）：度为0的结点；（度不为0的结点称为非终端结点或分支结点，分支结点也称为内部结点）

父结点（Parent）：有子树的结点是其子树的根结点的父结点；

子结点（Child）：结点的子树的根；

兄弟结点（Sibling）：具有同一父结点的各结点是彼此的兄弟结点；

路径和路径长度：从结点n1到nk的路径为一个结点序列n1,n2,...,nk,ni是n（i+1）的父结点；路径所包含的边的个数为路径的长度。

祖先结点（Ancestor）：沿树根到某一结点路径上所有的结点；

子孙结点（Descendant）：某一结点的子树中所有的结点；

堂兄弟结点：父结点在同一层的结点互为堂兄弟；

结点的层次（Level）：规定根结点在1层，其它任一结点的层数是其父结点的层数+1；

树的深度（Depth或高度）：树中所有结点的中最大层次是这棵树的深度；

有序树：将树中结点的各子树看成从左至右是有次序的（即不能互换）；否则为无序树；

森林（Forest）：m（m>0）棵互不相交的树的集合；

对树中的每个结点而言，其子树的集合即为森林，由此，也可以森林和树相互递归的定义来描述树；

二叉树（Binary Tree）：一个有穷的结点集合，特点是每个结点至多只有两颗树子树，二叉树的子树有左右之分不能任意颠倒；

由三个基本单元组成：根节点、左子树、右子树；

这个集合可以为空，若不为空，则它是由根结点和其左子树TL和右子树TR的两个不相交的子树组成；

特殊二叉树：斜二叉树、完美二叉树（满二叉树）、完全二叉树；

完全二叉树：有n个结点的二叉树，对树中结点按照从上至下、从左到右顺序进行编号，编号为i结点与满二叉树中编号为i结点在二叉树中位置相同；

二叉树的性质：

1.在二叉树的第i层上至多有2的i-1次方个结点（i>=1)；

2.深度为k的二叉树至多有2的k次方-1个结点（k>=1)；

3.对任何一颗二叉树T，如果其终端的结点数为n0，度为2的节点数为n2，则n0=n2+1；

4.具有n个结点的完全二叉树的的深度为[logn]+1；

5.如果对一颗有n个结点的完全二叉树的结点按层序编号，则对任一结点有:

①如果i=1，则结点i是二叉树的根，无父结点；如果i>1,则其父结点是结点[i/2]；

②如果2i>n，则结点i无左孩子；否则其左孩子结点是2i；

③如果2i+1>n，则结点i无右孩子；否则结点为2i+1；

二叉树的遍历：以一定的规则将二叉树中的结点排列成一个线性序列，得到二叉树中结点的先序序列或中序序列或后序序列；实质上是对一个非线性结构进行线性化操作，使每个结点（除第一个和最后一个外）在这些线性序 列中有且仅有一个直接前驱和直接后继；

先序遍历：若二叉树为空，则空操作；否则，

1.访问根节点；

2.先序遍历左子树；

3.先序遍历右子树；

中序遍历：若二叉树为空，则空操作；否则，

1.中序遍历左子树；

2.访问根节点；

3.中序遍历右子树；

后序遍历：若二叉树为空，则空操作；否则，

1.后序遍历左子树；

2.后序遍历右子树；

3.访问根节点；

层次遍历：需要建立一个循环队列；

线索链表：若结点有左子树，则其lchild域指示其左孩子，否则令lchild域指示其前驱；若结点有右子树，则其rchild域指示其右孩子，否则令rchild域指示其后继；为了避免混淆，尚需改变结点结构，增加两个标志域LTag 和RTag，以这种结点结构构成的二叉链表作为二叉树的存储结构，叫做线索链表，其中指向结点前驱和后继的线索；加上线索的二叉树称为线索二叉树；（中序线索二叉树上遍历，时间复杂度为O（n）；

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；

树的存储结构：链表结构

1.双亲表示方法：采用一组连续空间来存储每个结点，同时在每个结点中增设一个伪指针，指示其双结点在数组中的位置；根结点的下标为0，其伪指针域为-1；

该存储结构利用了每个结点（根结点除外）只有唯一双亲的性质，可以很快得到每个结点的双亲结点，但求结点的孩子需要遍历整个结构；

2.孩子表示法：将每个结点的孩子结点都用单链表链接起来形成一个线性结构，此时n个结点就有n个孩子链表（叶子结点的孩子链表为空链表）；

这种存储方式寻找子女的操作非常直接，而寻找双亲的操作需要遍历n个结点中的孩子链表指针域所指向的n个孩子链表；

3.孩子兄弟表示法：二叉树表示法，以二叉树的结构作为树的存储结构；

每个结点包括三个部分内容：结点值、指向结点第一个孩子结点的指针、指向结点下一个兄弟结点的指针（沿此域可以找到结点的所有兄弟指针）；

树与二叉树的转换：每个结点左指针指向它的第一个孩子，右指针指向它在树中的相邻右兄弟；根节点没有兄弟，所以对应的二叉树没有右子树；

树的遍历：

先根遍历：若树非空，先访问根结点，再依次遍历根节点的每颗子树，遍历子树时仍遵循先根再子树的规则；（对应二叉树的先序遍历）

后根遍历：若树非空，先依次遍历根节点的每颗子树，再访问根节点，遍历子树时仍遵循先子树后根的规则；（对应二叉树的中序遍历）

森林的遍历：

先序遍历：若树非空，则进行下列操作（对应二叉树的先序遍历）：

1.访问森林第一棵树的根节点；

2先序遍历第一棵树根节点的子树森林；

3.先序遍历除去第一棵树之后剩余的树构成的森林；

中序遍历：若树非空，则进行下列操作（对应二叉树的中序遍历）：

1.中序遍历森林中第一颗二叉树的根结点的子树森林；

2.访问第一棵树的根节点；

3.中序遍历除去第一棵树之后的剩余数构成的森林；

赫夫曼树（Huffman）： 又称最优树，是一类带权路径长度最短的树；

带权路径长度：树中所有叶结点的带权路径长度之和；记为WPL = 和(Wi\*Li)

Wi:第i个叶结点所带的权值

Li:该结点到根结点的路径长度

在含有n个结点的二叉树中，其中带权路径长度（WPL）最小的二叉树称为赫夫曼树 ，也称最优二叉树；

赫夫曼树的构造：

1.将n个结点分别作为n棵仅含一个结点的二叉树，构成森林F；

2.构造一个新的结点，从F中选取两棵结点权值最小的树作为新结点的左右子树，并将新结点的权值置为左右子树上根结点的权值之和；

3.从F中删除刚选的两棵树，同时将新得得树加入F中；

4.重复步骤2和3，直到F中只剩下一颗树为止；

赫夫曼树得特点：

1.每个初始结点最终都成为叶结点，且权值越小的结点到根结点的路径长度越大；

2.构造过程中共建立了n-1个结点，因此赫夫曼树的结点总数为2n-1；

3.每次构造都选择2棵树作为新结点的子树，因此赫夫曼树不存在度为1的结点；

要设计长短不等编码，则必须是任一字符的编码都不是另一个字符的编码的前缀，这种编码叫做前缀编码；

赫夫曼编码：设计电文总长的最短的二进制前缀编码即为以n种字符出现的频率作为权，设计一颗赫夫曼树，由此得到二进制前缀的编码便称为赫夫曼编码；

固定长度编码：对每个字符用相等长度的二进制位表示；

可变长度编码：允许对不同字符用不等长的二进制位表示；