树

逻辑非线性结构：

数据间关系：1：n

一个节点有且金由一个前驱节点，可以由多个后继；

没有任何节点的树成为空树；

在树的众多节点中，有且仅有一个节点无前驱节点，这个节点称为根节点（Root）；

树的节点中，没有后继节点称为叶子节点（Leaf）

术语：

1、根结点

2、叶子节点；

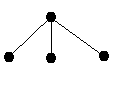
3、树的高度（层数）；

4、节点扇出度（有几个子节点）;

5、树的宽度。

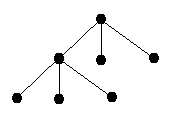
6、树的度：树中最大度节点的度数

树的形式化语言表示方法：

树中节点个数的计算:

度为3，总结点数为4；

度为n，总结点个数为n+1



度 个数

3 2

总节点个数为2\*(3+1) -1

Σ个数\*（度+1）

将所有度非0的节点（除根结点外）都重复了一次；减去度非0的节点总数再+1

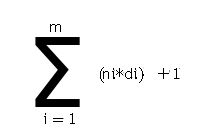
设度为d1到dm的节点个数分别为n1到nm个，则：

n总 =

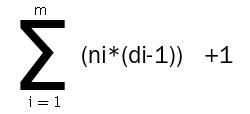
=n1\*(d1+1)+n2\*(d2+1)+...+nm\*(dm+1) - (n1+n2+...+nm) +1

=n1\*d1 + n1 + n2\*d2 +n2 +...+nm\*dm +nm-n1-n2-...-nm+1

=n1\*d1+n2\*d2+...+nm\*dm+1



=



n0=

树的表示方法：

1、链表方式：

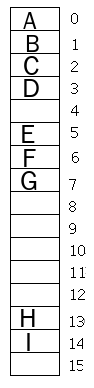
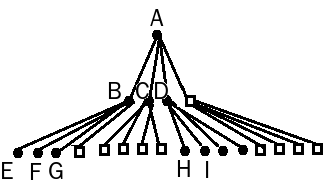
typedef struct TREE{

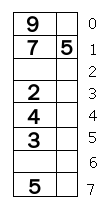
USER\_TYPE data;

struct TREE\*children;

} TREE;

2、数组方式：



3、数组链表：

下标 数据 父节点下标

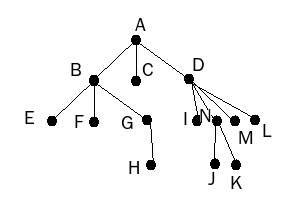
0 A -1

1 B 0

2 C 0

3 D 0

4 E 1



树的遍历：

广度优先遍历：

A，B，C，D，E，F，G，I，N，M，L，H，J，K

深度优先遍历：

A，B，E，F，G，H，C，D，I，N，J，K，M，L

队列：A

A

A出队列，将A的所有子节点入队列：

队列：B，C，D

B

B出队列，将B的所有子节点入队列

队列：C，D，E，F，G

将C出队列，将C的所有子节点入队列

C

队列：D，E，F，G

…

得到的遍历结果为：

A，B，C，D，E，F，G，I，N，M，L，H，J，K（广度优先遍历）

堆栈：将A入栈

A

访问A，将A出栈，将A的所有子节点入栈

堆栈（自上而下）：B，C，D

访问B，将B出栈，将B的所有子节点入栈

B

堆栈：E，F，G，C，D

E

将E出栈，E的所有子节点入栈：

……

得到的访问结果为

A，B，E，F，G，H，C，D，I，N，J，K，M，L（深度优先遍历）

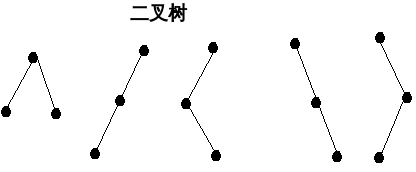
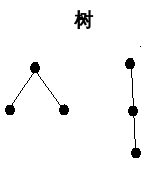
广度优先遍历使用队列；深度优先遍历使用堆栈。

二叉树

二叉树不是度为2的树。

二叉树与度为2的树的差别：

设有三个不分顺序的节点，要求用他们（不重复使用）组成形态不同的树和二叉树，问，能够组成的树和二叉树的种类分别是多少个？



满二叉树

高度为h的满二叉树中，共有2^h-1个节点。

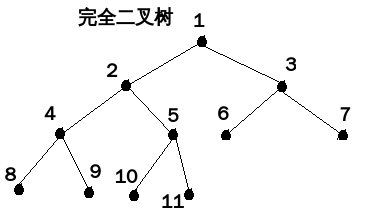
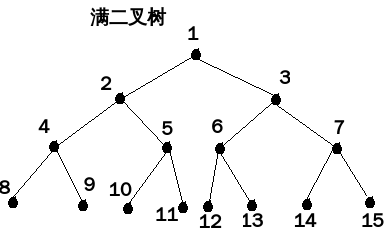
满二叉树第i层节点数量为：2^(i-1)

拥有n个节点的满二叉树的高度为：h = log2(n+1)

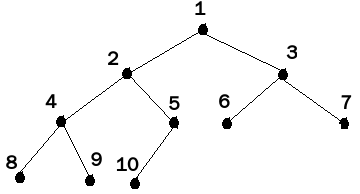
完全二叉树

对于一颗高度为h的满二叉树，从上至下，从左至右，依次从1开始编号；

对满二叉树中的节点作删除工作，删除若干个节点，使得剩余节点的编号满足如下条件：

从1开始编号，连续编号。

由于完全二叉树按照自上至下，自左至右，节点的编号是连续的，因此，可以将其存储到一个连续存储空间中。且，存在如下性质：



设完全二叉树的节点个数为n

下标从1开始

最后一个非叶子节点的下标：n/2

设i取值范围[1, n/2]

下标为i的节点的左、右孩子节点，其下标分别是：2i和2i+1（2i+1<=n）

若2i+1>n，则说明：这棵完全二叉树有且仅有一个度为1的节点，这个节点的性质：

1、其下标是n/2;

2、它只有一个左孩子。

二叉树的一个重要结论：

n0 = n2 + 1

这个性质无论对于完全二叉树还是满二叉树都是可行的，因为二叉树也满足度为2的树的一般性质，即：

n0 = (1-1)\*n1 + (2-1)\*n2 + 1

= n2 + 1

设一颗完全二叉树的节点总数是n，且n是偶数；问该树的叶子节点数目？

n总 = n0 + n1 + n2

= n0 + 1 + n0 – 1

= 2n0

n0 = n总/2

若n总是奇数

n总 = n0 + n1 + n2

= n0 + 0 + n0 – 1

= 2n0-1

n0 = (n总+1)/2

高度为h的完全二叉树节点总数取值范围：

[2^(h-1), 2^h-1]

二叉树的存储结构：

二叉树的线性存储结构，将一颗普通二叉树，通过增加虚节点使之成为完全二叉树后，存储在线性存储空间中。

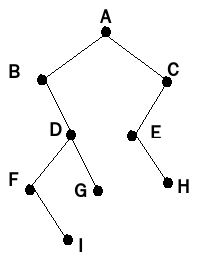
二叉树非线性存储结构（经典存储结构）

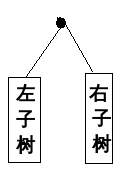
typedef struct BTREE{

USET\_TYPE data;

struct BTREE \*leftChild;

struct BTREE \*rightChild;

}BTREE;



二叉树节点的遍历

先根序：先根，再左子树，再右子树

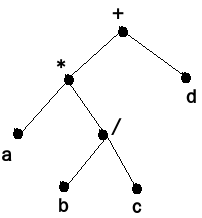
中根序：先左子树，再根，再右子树

后根序：先左子树，再右子树，再根

先根序：A，B，D，F，I，G，C，E，H

中根序：B，F，I，D，G，A，E，H，C

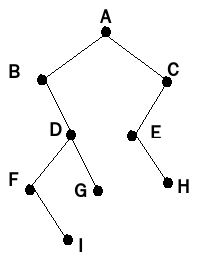
后根序：I，F，G，D，B，H，E，C，A



a \* b / c + d（中波兰式）

+ \* a / b c d（先波兰式）

a b c / \* d +（后波兰式）



定义一个输入二叉树的规范式：

Root(L, R)

Root(L)

Root(, R)

A(B(,D(F(,I),G)),C(E(,H)))

错误的写法：

A(B(E),C(H),D)

BEGIN: ALPHA | END

ALPHA: ( | , | ) | END

( : ALPHA | ,

) : , | ) | END

, : ALPHA

END

状态：

BEGIN；

ALPHA；

LEFT\_BRACKET;

RIGTHT\_BRACKET;

COMMA;

END.

节点处理：

1、产生一个节点；

分3种情况：

1）root为NULL；

2）root不为NULL，但是栈为空，一种错误

3）root不为NULL，栈也不为空

<3.1>左右孩子标志为左孩子，则，将栈顶元素（指针）所指向的节点的左孩子指针，指向新节点；

<3.2>若栈顶指针所指向的节点的右孩子指针非空，这是另一种错误（出现了第三个孩子）

左括号：

当前节点首地址入栈；更改左右孩子标志

右括号：

栈顶指针出栈

逗号：

更改标志为右孩子