### 树

1. 树是非线性结构，树结构中，节点之间的关系是前驱唯一后继不唯一，节点之间是多对多的关系
2. 树的相关术语

B

L

M

K

I

J

H

G

E

F

D

C

A

1

2

3

4

①A为根节点

②BCD为A的孩子节点

③A是BCD的双亲节点

④HIJ为兄弟节点

⑤EGH为堂兄弟节点，他们的双亲节点为兄弟节点或者是堂兄弟节点

⑥祖先一个节点的节点：从根节点待该节点路径上的所有节点，K的祖先节点为ABE

⑦子孙节点：一个节点的直接后继或者间接后继，D的子孙节点为HIJM

⑧前辈：层号小于该节点层号的节点都是该节点的前辈

⑨后辈：层号大于该节点层号的节点都是该节点的后辈

⑩节点的度：一个节点的子树个数

⑪叶节点：度为零的节点，即没有后继节点，也称终端节点

⑫分支节点：度不为零的节点，也称非终端节点

⑬节点的层次：从根节点开始定义，根节点的层次为1，根节点的直接后继节点的层次为2，以此类推，上图树层次为4层

⑭树的度：树中所有节点的度的最大值

⑮树的高度:树中所有节点层次的最大值

⑯同构：结构相同

### 二叉树

1. 二叉树满足的两个条件：

①每个节点的度都不大于2

②每个节点的孩子节点次序不能任意颠倒

每个节点只能有0,1或2节点，每个孩子有左右之分

1. 二叉树的性质：

①二叉树中，每层最多有2^(i-1)个节点

②深度为k的二叉树其节点总数最多有(2^k) -1

③对于一颗任意二叉树，若终端节点个数为n0，其度数为二的节点个数为n2，则n0=n2+1

满二叉树：深度为k且有(2^k) -1个节点，即每层都是满的，每层都具有最大节点个数

完全二叉树：深度为k节点数为n，其节点1~n的位置号与其等高的满二叉树1~n位置一一对应，则为完全二叉树

满二叉树一定是完全二叉树，完全二叉树不一定是满二叉树

1

满二叉树

3

2

6

4

7

5

1

完全二叉树

3

2

6

5

4

④具有n个节点的完全二叉树深度为[log2^n]+1

⑤.对于具有n个节点的完全二叉树，如果按照从上到下从左到右的顺序对对二叉树中的所有节点从1

开始按顺序编号，则对于任意的序号为i的节点有：

·如果i=1，则该节点就是根节点，无双亲节点，如果i>1，则序号为i的节点的双亲节点序号为[i/2]

·如果2i>n，则序号为i的节点无左孩子，如果2i<=n，则序号为i的节点的左孩子节点序号为2i

·如果2i+1>n则序号为i的节点无右孩子，如果2i+1<=n，则序号为i的节点的右孩子节点序号为2i+1

1. 二叉树的存储结构：二叉树的结构是非线性的，每个节点最多可有2个后继，二叉树的存储结构分为两种--顺序存储结构和链式存储结构

①顺序存储结构对于完全二叉树来说，

可以将其数据元素逐层放到一组连续的存储单元中，用一维数组做存储结构，将二叉树中编号为i的节点存放在数组的第i个位置中，根据二叉树的性质5，节点i的左孩子的位置为2i，右孩子位置为2i+1；但是对于一般二叉树来说，必须用虚节点将其补成一颗完全二叉树，但是这样就会造成空间的浪费，比如，对于一颗深度为k的二叉树，在最坏的情况下（每个节点只有右孩子）需要使用2^k -1个存储单元，而实际该二叉树只有k个节点，就会造成空间的浪费

完全二叉树顺序存储结构：

1

2 3

4 5 6

1 2 3 4 5 6

一般二叉树顺序存储结构：

A

B

D

C

A ^ B ^ ^ ^ C ^ ^ ^ ^ ^ ^ D

②链式存储结构：对于任意的二叉树来说，每个节点只有一个双亲节点（根节点除外），最多有两个孩子，可以设计每个节点至少包含三个域：数据域，左孩子域，右孩子域

节点结构：

^ G ^

^ E ^

^ F ^

^ D

C

A

B ^

F

E

H

D

C

B

A3

LChild Data RChild

\*如果一个二叉树含有n个节点，那么二叉链表一定有2n个指针域，其中必定有n+1个空的链域

4.二叉树的遍历与线索化：按照一定规律对二叉树中的每个节点进行访问且仅访问一次

遍历过程是个递归过程

①先序遍历（DLR）

二叉树不为空就进行如下操作

·先访问根节点

·按先序遍历左子树

·按先序遍历右子树

②中序遍历（LDR）

二叉树不为空就进行如下操作

·先中序遍历左子树

·访问根节点

·按中序遍历右子树

③后序遍历（LRD）

二叉树不为空就进行如下操作

·按后序遍历左子树

·按后序遍历右子树

·访问根节点

根据父节点的输出位置就可以判断是什么遍历