三、树Tree（上）

1.插入、删除、查找

1.1.查找Searching：给定关键字K，从集合R中找出与K相同的记录

静态查找：没有插入和删除操作

动态查找：可能发生插入和删除

1.2.二分查找Binary Search：

数据有序，且联系存放（数组），链表不行

left；right；mid

1.3.判定树：

每个结点需要查找的次数刚好为该节点所在的层数

查找次数不会超过判定树的深度

n个结点判定树的深度为[logn]+1

平均查找长度ASL:average searching length

2.定义：

根Root：r

子树SubTree：互不相交（若相交则为非树）

除了根节点，每个结点仅有一个父节点

一棵N个结点的树有N-1条边（root没有连向根结点的边）

树是保证结点联通的边最少的连接方式

3.基本术语

结点的度Degree：结点子树个数（等于子节点个数）

树的度：树所有节点中最大的度树

叶节点：度为0的结点

父节点Parent：有子树的节点

子节点：Child

兄弟结点：Sibling有相同父节点

路径和路径长度：路径所包含边的个数

祖先结点Ancestor：从root到某结点路径上的所有结点

子孙结点Descendant

结点的层次Level：root在第一层

树的深度Depth：树所有节点中最大层次

4.树的表示：儿子-兄弟表示法

Element

FirstChild NextSibling

5.二叉树：度为2的树（上图儿子兄弟表示法旋转45度）

Element

Left Right

5.1.二叉树T：root+TL左子树+TR右子树

斜二叉树Skewed Binary Tree：只有左/右子树

完美二叉树Perfect BT/满二叉树Full BT：每个结点左右都有

完全二叉树Complete BT：从上至下从左至右编号，与完美二叉树编号相同（允许满二叉树最后一行右边有缺）

5.2.重要性质

第i层最多有2^(i-1)个节点

深度为k的二叉树最大结点总数为2^k -1,k>=1

叶节点个数等于度为2的结点个数加一：n0=n2+1

5.3.重要操作：

遍历Traversal：

先序遍历PreOrderT：根左右：第一次碰到就pop

中序遍历InOrderT：左根右：第二次碰到pop

后序PostOrderT：左右根：第三次碰到pop

层次遍历LevelOrderT：从上到下从左到右

5.4.存储结构：

5.4.1.顺序存储结构：用数组存储

a.完全二叉树：从上至下从左至右顺序存储

非根节点父节点序号是[i/2]向下取整

结点左孩子序号：2i

结点右孩子序号：2i+1

大于n则没有左/右孩子

b.一般二叉树：

补齐成对应的完全二叉树再存

会造成空间浪费

5.4.2.链表存储结构：

data+左指针+右指针

5.5.遍历：

a.递归实现

先序中序后序遍历经过结点路线一样，但访问结点时机不同

b.非递归算法（e.g.中序遍历）:使用堆栈

有元素put，无元素pop

【图】

c.核心问题：二维结构的线性化

d.层序遍历（使用队列）：根节点出队，其左右儿子入队

e.两种遍历序列可以唯一确定二叉树（必须包含中序）（先确定根结点）