二叉树（英语：Binary tree）是每个节点最多有两个子树的树结构。通常子树被称作“左子树”（*left subtree*）和“右子树”（*right subtree*）

二叉树的每个节点至多只有二棵子树(不存在度大于2的节点)，二叉树的子树有左右之分，次序不能颠倒。

概念:

1. 树的节点的度数:此节点所含子树的个数(二叉树的节点的度数最大为2).
2. 树的度数:最大的节点的度(二叉树的度数最大为2).
3. 叶子节点:树底部的节点拥有空子树被称为“叶子”节点(度为零的节点).
4. 二叉树的层:根节点为第1层.(一般根节点被定义为第1层,也有教材定义为第0层)
5. 二叉树的深度/高度:二叉树的最大层数.(根节点定义为第1层,深度==最高层)
6. 满二叉树:所有叶节点都在最底层的完全二叉树(若深度为k,则节点个数为2^k-1个).
7. 完全二叉树:只有最下面的两层结点度能够小于2，并且最下面一层的结点都集中在该层最左边的若干位置的二叉树(深度为k的 二叉树,第1层到k-1层的节点都达到最大值,且第k层所有的节点从左到右连续紧密排列)
8. 二叉搜索树|二叉查找树|二叉排序树:(Binary Search Tree[ordered binary tree])根节点的左子树所有节点的值小于等于根节点的值, 根节点的右子树所有节点的值大于根节点的值
9. 线索二叉树(引线二叉树) : 所有应该为空的右孩子指针指向该节点在中序序列的后继，所有应该为空的左孩子指针指向该节点的中序序列的前驱。(在N个节点的二叉树中，每个节点有2个指针，所以一共有2N个指针，除了根节点以外每一个节点都有一个指针从它的父节点指向它，所以一共使用了N-1个指针。所以剩下2N-(N-1)个空指针。)
10. 平衡二叉树(Balanced Binary Tree)：是一种结构平衡的二叉搜索树，即叶节点深度差不超过1，它能在O(logn)内完成插入、查找和删除操作(常见的有: 1.AVL树 2.红黑树 3. Treap)

11. 树的遍历种类: 1)深度优先遍历(先序,中序,后序).2)广度优先遍历.

12. 普通树到二叉树的转换: 1)树的所有兄弟节点(父节点相同)之间加一根线. 2)对每个节点, 除了保留与长子(最左孩子)的连线外, 去掉所有该节点与其他的孩子节点间的连线. -> 完成

13. 节点的路径长度: 从根节点到该节点的路径上连接数

性质:

1. 二叉树的深度k==二叉树的最大层树
2. 二叉树的第i层至多有2^(i-1)个节点.
3. 深度为k的二叉树至多共有2^k-1个节点.
4. 对任何一棵二叉树T，如果其终端节点数为n0,度为2的节点数为n2,则n0=n2+1.
5. 元素个数为n的二叉树,

Things To Understand

1. The node/pointer structure that makes up the tree and the code that manipulates it

   树是由节点/指针结构构成的,用代码操作这棵树

2. The algorithm, typically recursive, that iterates over the tree

   算法，通常是递归，遍历树