1 树是 n 个节点的有限集合,st 有一个根节点,其他节点分成 m 个不相交的集合,也都是树,称作这个根的子树

2 逻辑结构:有且只有一个节点没有前驱,称作树的根,除它外,每一个节点都只有一个前驱,路径

3 嵌套括号表示法:(根(一个子树)(又一个子树))

4 层数,高度,深度

5 有序树:子节点从左到右编号,但度为 2 的有序树并不是二叉树,必须子节点严格区分左右,否则第二子节点可能会自然顶替成为第一子节点

6 森林:多棵不相交的树的(有序)集合

7 森林与二叉树的一一对应,左子节点是第一个子节点,右子节点是下一个兄弟

8 先根次序:访问根节点->从左到右先根遍历每一棵子树

9 后根次序:从左到右后根遍历每一棵子树->访问根节点

10 先根周游等于对应二叉树的前序周游　后根周游相当于中序周游

11 广度优先

12 链式存储

a) 子节点表示法　节点**数组**,父节点,孩子链表　查孩子个数,结点值容易,找兄弟结点困难　归并容易
b) 指针数组法　结点,度数,子节点指针的数组
c) 指针链表法　结点,孩子指针的链表
d) 静态左子右兄表示法　结点**数组**,左子节点,节点值,父节点,右兄弟结点
比子节点表示法空间效率更高,每一个结点仅需要固定大小的空间
归并树只需要设置指针的值
e) 动态左子右兄 二叉链表表示法 用二叉树替换树 更为常见

13 查找父节点->广度优先实现,临时保存父节点,子节点进栈,直到找到要找的结点

14 删除树->后序遍历进行删除 删除结点->如果是最左子节点,父亲接到它的右兄弟,不是,左兄弟接到右兄弟

15 父指针表示法:指示父节点的位置,可以唯一的表示一棵树

16 寻找父节点常数时间,求树根结点方便,寻找兄弟结点麻烦,适合于无序树的情况

17 并查集:求解等价类问题

18 重量权衡合并原则:结点较少的根节点指向结点较多的根节点,整体深度限制在 O(log n)
(每一次归并高度最多加一,结点个数成倍增加)

19 路径压缩算法:路上遇到的结点都指向根节点

20 复杂度:space O(n)　find / union O(a(n)) ackerman(反)函数,增长非常缓慢的函数,可以认为是小于 5

21 顺序存储

a) 带右链的先根次序 ltag(有孩子为 0,没孩子为 1)　rlink 指向下一个兄弟
b) 带双标记位的先根次序 ltag(同上)　rtag(有兄弟为 0,没有为 1)
i. 扫描到一个 rtag 为 0 的点进栈,扫描到一个 ltag 为 1 的点,弹栈,下一个结点是它兄弟
ii. 处理最后一个结点,孩子兄弟都为空
c) 带度数的后跟次序表示法 按后跟次序顺序存储,结点包括结点值,结点的度数
从左到右扫描,遇到 0 度结点入栈,k 度结点就弹出 k 个节点做他儿子,把它入栈
d) 带度数的先根次序
e) 带度数的层次次序

f) 带双标记的层次次序 ltag,rtag

Ltag 为 1,llink 置空,ltag 为 0,结点入队列, rtag 为零,后面的结点就是右兄弟

如果 rtag 为 0,出队列,下一个结点是他儿子

22 K 叉树:每一个结点都有 k 个有序子节点,完全 k 叉树,满 k 叉树