$1, 2, 3, ...n...., 9, 11...$
n 出栈,就说明前面的元素已入过栈,n 后面的元素部分可能未入栈

1.树的基本定义

零个节点的树为空树

....

2.树的存储结构

1.顺序存储(并查集)

数组对应编号下标元素中存储父结点编号
重点记录了节点之间的亲子关系

2.链式存储(二叉链表)

- 孩子存储(亲子存储)
- 孩子兄弟表示法
  左指针指向长子,右指针指兄弟

二叉树

- 斜树:每层只有一个节点,就是链表,节点数和树的深度相同
- 满二叉树:非叶子节点的度都为二,相同深度,节点数最多,叶子数也最多
- 完全二叉树:从右向左删除部分节点
  - 1.节点编号对应相应满二叉树
  - 2.叶子节点只出现在最下两层,且底部叶子节点只出现在左侧连续部分
-
  - 3.如果存在度为一的节点,则只有左孩子
-
  - 4.同节点数二叉树,完全二叉树深度最小

二叉树的性质(主要是选择题计算)

满二叉树节点编号之间的关系

二叉树存储结构

1.链式(考过节点定义代码)

```
typedef struct BTNode{
int key;//data
BTNode* lchild;
BTNode* rchild;

}BTNode;

int n[max];//n 数组
n[cengshu]++;
```

```
printf("%d",p->data);
```

.遍历算法(DFS)

- 先序:根左右
- 中序:左根右
- 后序:左右根

pre:ABDECFG
mid:DBEAFCG
post:DEBFGCA

> 图的 BFS

if(...) prorder(..);剪枝法,递归进入之前先判断;

循环和栈结合的遍历算法

入栈顺序与遍历顺序相反

后序遍历

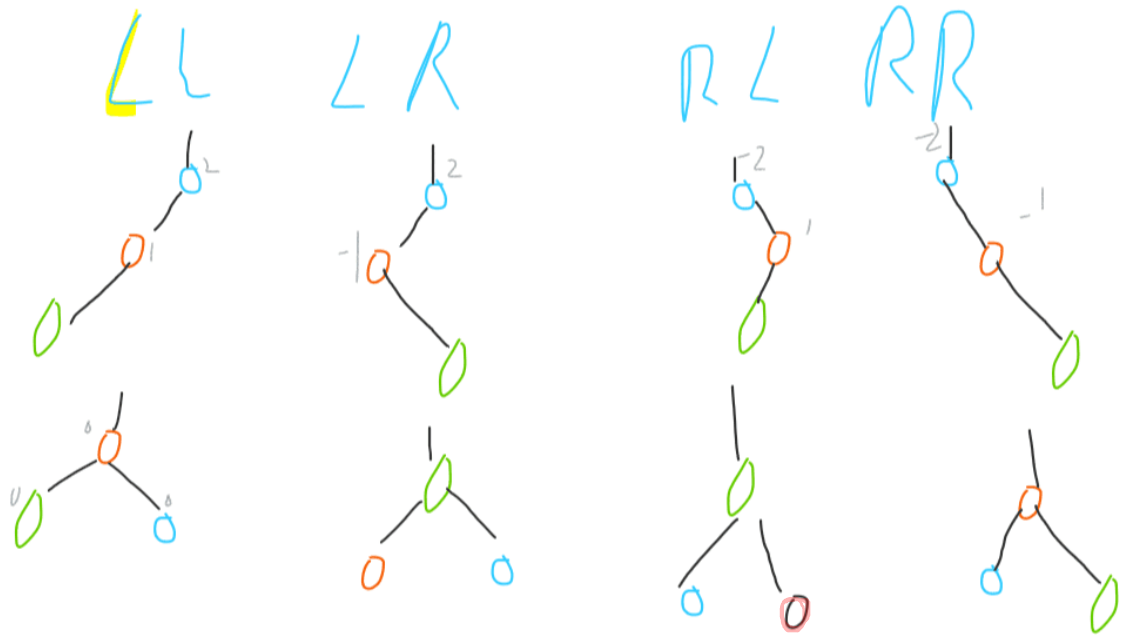
先实现逆后序遍历,再倒着输出(双栈法)
逆后序遍历:先序遍历中左孩进栈,右孩子入栈

sibling

4.构造过程中共新建$N-1$个节点,节点总数为$2N-1$

郝夫曼树和编码判定:

1. 任何编码都不是别的编码的前缀
2. 树中起码不包括单分支节点



1. 含有 n 个节点的平衡二叉树的最大深度为 $log_2 N$,平均查找长度就是 $O(log_2 N)$
2. 假设以 N_h代表深度为h的平衡二叉树中含有的最少节点数,则,

$$N_h = N_{h-1} + N_{h-2} + 1$$

$$N_0 = 0, N_1 = 1, N_2 = 2$$