# Tree

## 树的遍历

1、题号

1. 94
2. 102
3. 103
4. 107
5. 116
6. 117

2、解析

1. 对于常规的前序、中续、后续遍历：前序3种，中续3种，后续5种
2. 层遍历：队列

## 给定序列求可生成的搜索二叉树的总数

1、题号

1. 95
2. 96

## 树特性的判定

1、题号

1. 98
2. 110

2、解析

1. 判定是否满足搜索二叉树：递归时带入区间范围即可
2. 判定是否平衡二叉树：采用从底向上的方法，递归函数返回给定节点的最大深度，然后在函数体内维护一个全局布尔值即可

## 恢复搜索二叉树(某一对节点被调换了位置)

1、题号

1. 99

2、解析

1. 从中序遍历来考虑
2. 一种直观的思路就是得到中序遍历的一个List，然后找到两个节点
3. 空间复杂度更小的方法是：找到val(i)>val(i+1)的情况两次，这两个错误节点就找到了

## 判断两棵树是否相同，判断一棵树是否对称

1、题号

1. 100
2. 101

2、解析

1. 递归，注意递归边界条件就行

## 树的深度

1、题号

1. 104
2. 111

2、解析

1. 自顶向下的方法，给递归函数传入一个目前的深度值即可

## 构建二叉树

1、题号

1. 105
2. 106
3. 108

2、解析

1. 根据前序中序或者中序后序来构建搜索二叉树

* 前序给定一个区间(必须是一个子树的区间，那么根节点一定是这个区间的左边界)
* 后序给定一个区间(必须是一个子树的区间，那么根节点一定是这个区间的右边界)
* 用中序来判断左右子树的大小
* 通过一个map映射value--->pos

1. 根据有序数组或链表构建二叉树

## 路径和

1、题号

1. 112
2. 113

2、解析

1. 求所有路径和，或者是否有指定路径和，自顶向下比较好，因为不能过早得筛选信息
2. 求最大最小路径和，自顶向下或者自底向上都可以试试

## 转为链表树

1、题号

1. 114

2、解析

1. 对于以指定节点为根节点的子树，其Linked List由该子树的左子树和右子树组成。于是可以采用自底向上的方式来进行连接，递归函数返回Linked List的头节点和尾节点
2. 或者采用前序遍历的方式连接