一. 树:

① 遍历: DFS (前. 中. 后序遍历)

实现方式: 递归 或栈

算顺序(第k个).

递归:
```
def dfs(root)
    if not root or self.k <=0: return
    dfs(root.right)
    self.k -=1
    if self.k==0: return root.val
    dfs(root.left)
```

栈: (后序遍历)
```
stack, p = [], root
while stack or p:
    while p:
        stack.append(p)
        p=p.right
    if stack:
        temp = stack.pop()
        k-=1
        if k==0: return temp.val
        p= temp.left
```

算高度
```
函数(root):
    l= 函数(root.left)
    r = 函数(root.right)
    return max(l,r) +1
```

遍历打印:
```
函数(root.left)      前
函数(root.right)     中
print(root.val)      后
```

BFS (层序遍历)

实现方式: 往往利用队列实现

temp[] 存下一层结点

count 指层数

stack [] 存当前结点

```
def 函数(root):
    if root == None: return 0
    count = 0
    stack = [root]
```

```
while (stack):
    temp = []
    for cur in stack:
        temp 入左子树, 入右子树
    stack = temp
    count += 1
```

求路径和.

动态规划

if (root.val == sum)
++

- 从根节点出发: 递归. += (root.left, sum - root.val) + root.right, sum - root.val)
- 不从根节点出发: 函数(root.left, sum) + 函数(root.right, sum).

二叉搜索树: 左 < 中 < 右.

二叉搜索树的中序遍历得到的序列一定为**升序**