# 二叉树



## 二叉树的性质

1. 二叉树的第i层至多有2^(i - 1)个节点。
2. 深度为k的二叉树至多有2^k - 1个节点。
3. 深度为k，且节点数量为2^k - 1的二叉树为满二叉树。

type TreeNode struct {

Val int

Left \*TreeNode

Right \*TreeNode

}

## 前序遍历

根->左->右

1 2 4 5 7 8 3 6

func preOderTraverse(root \*TreeNode) {

if nil != root {

fmt.Println(root.Val, “ ”)

preOderTraverse(root.Left)

preOderTraverse(root.Right)

}

}

## 中序遍历

左->根->右

4 2 7 5 8 1 3 6

func midOderTraverse(root \*TreeNode) {

if nil != root {

midOderTraverse (root.Left)

fmt.Println(root.Val, “ ”)

midOderTraverse (root.Right)

}

}

## 后序遍历

左->右->根

4 7 8 5 2 6 3 1

func postOderTraverse(root \*TreeNode) {

if nil != root {

postOderTraverse(root.Left)

postOderTraverse(root.Right)

fmt.Println(root.Val, “ ”)

}

}

## 层次遍历

func breadthTraverse(root \*TreeNode) {

}

## 二叉树深度

func treeDepth(root \*TreeNode) int {

if nil == root {

return 0

}

return int(math.Max(float64(treeDepth(root.Left)), float64(treeDepth(root.Right)))) + 1

}

## 二叉树节点个数

func treeNodeNum(root \*TreeNode) int {

if nil == root {

return 0

}

return treeNodeNum(root.Left) + treeNodeNum(root.Right) + 1

}

## 二叉树第K层的节点个数

求以root为根的k层节点数目，等价于求root左节点为根的第k-1层和root右节点为根的第k-1层的节点数目之和。

func KthLvNodeNum(root \*TreeNode, k int) int {

if nil == root {

return 0

}

if 1 == k {

return 1

}

return KthLvNodeNum(root.Left, k - 1) + KthLvNodeNum(root.Right, k - 1)

}

## 判断两个二叉树是否相同

func isSameTree(rootA \*TreeNode, rootB \*TreeNode) bool {

if nil == rootA && nil == rootB {

return true

}

if (nil == rootA && nil != rootB) || (nil != rootA && nil != rootB) {

return false

}

if (rootA.Val != rootB.Val) {

return false

}

return isSameTree(rootA.Left, rootB.Left) && isSameTree(rootA.Right, rootB.Right)

}

## 判断是否平衡二叉树

# 二叉查找（搜索）树

对于树中任意一个节点X，其左子树中任意节点的值都小于X的值，其右子树中任意节点的值都大于X。

# 平衡二叉树（AVL树）

空树或它的左右两个子树的高度差的绝对值不超过1，并且左右两个子树都是一棵平衡二叉树。

## 判断是否为平衡二叉树

func isAVLTree(root \*TreeNode) bool {

if nil == root {

return true

}

if int(math.Abs(float64(treeDepth(root.Left) - treeDepth(root.Right)))) > 1 {

return false

}

return isAVLTree(root.Left) && isAVLTree(root.Right)

}

# 完全二叉树

# 红黑树