中心主题

分支主题 10

自由主题

33. 滑雪场

```
func dfs(level, pair, left, right int, s string) {

    // terminator
    if level == 7 {pair 2
        res = appendRes, s)
        return
    }
    // process current level
    if left down {
    if left < pair {
        dfs(level+1, pair, left+1, right, s+"(")
    }
    if right < left {
        dfs(level+1, pair, left, right+1, s+")")
    }
    // reverse the current level status if needed

    }

```

226. 翻转二叉树

```
    // process current level
    if root != nil && root.Left != nil {root.Right, root.Left
    // drill down
    invertTree(root.Left)
    invertTree(root.Right)
}
```

90. 翻转图的二叉树

```
func invertTree(root *TreeNode) {
    if root != nil && root2 == nil {
    return true
    }
    if root != nil || root2 == nil {
    return false
    }
    if root.Val != root2.Val {
    return false
    }
    return (fIsSyllabusInvertTree(root.Left, root2.Left) &&
    fIsSyllabusInvertTree(root.Right, root2.Right) &&
    fIsSyllabusInvertTree(root.Left, root2.Right) &&
    fIsSyllabusInvertTree(root.Right, root2.Left))
}

```

98. 验证二叉树

```
func dfs(root *TreeNode, low, high int) {
    // process
    // terminator
    if root == nil {
        return true
    }
    // process
    if root.Val >= high || root.Val <= low {
        return false
    }
    // drill down
    return dfs(root.Left, low, root.Val) && dfs(
    root.Right, root.Val, high)
}

```

234. 二叉树的公共祖先

```
// 当前已找到的公共祖先个数初始值设为0，
// 递归遍历根结点，
// 对于一个遍历到的结点root，如果root等于p或q 则
// 当前已找到的公共祖先个数加1，
func lowestCommonAncestor(p, q, root *TreeNode) {
    // 递归遍历左子树，当前已找到的公共祖先个数
    // 如果已经找到一个，则直接返回这一个，
    // 否则看左右子树的情况，若都找到了，
    // 则当前结点就是最近公共祖先，
    // 否则返回当前已找到的公共祖先个数
}

func lowestCommonAncestor(root, p, q *
TreeNode) {
    if root == nil || root.Val == p.Val || root.
    Val == q.Val {
        return root
    }
    leftAncestor = lowestCommonAncestor(root.
    Left, p, q)
    rightAncestor = lowestCommonAncestor(root.
    Right, p, q)
    if leftAncestor != nil && rightAncestor !=
    nil {
        return nil
    }
    if leftAncestor != nil {
        return rightAncestor
    }
    if rightAncestor != nil {
        return leftAncestor
    }
    return root
}

```

105. 从前序与中序遍历序列构造二叉树

```
func buildTree(preorder, inorder []int) {
    var  preIndexSt[]
    Left, buildTreeIterative(preIn+1, inorder[i]-1,
    preIn+1, buildTreeIterative(preIn+1, inorder[i+
    1,
}

```

进行

22. 括号

```
func dfs(n, s[]byte int, path [][]
    // terminator
    if len(path) == n {
    tmp := make([]int, lc
    copy(tmp, path)
    res = appendRes, tmp)
    return
    }
    // process
    // 定义数组保存结果集，长度为2*n
    // 定义两个变量，一个表示左括号，一个表示右括号
    // 当一个括号的个数小于n时，
    // 则可以继续往这一个方向进行递归，
    // 当左括号个数等于右括号个数时，
    // 则只能往这一个方向进行递归，
    // 当两个括号个数相等时，则结束递归。
    // 当两个括号个数等于n时，则结束递归。
    // 当两个括号个数等于n时，则结束递归。
    // 当两个括号个数等于n
```