# 【第五课】二叉树与经典问题

## 144. 二叉树的前序遍历

前序遍历就是按根节点，左子节点，右子节点来输出整个树。我们只需要递归的遍历树的左子树和右子树就可以了。

递归写法：

```
/**
 * Definition for a binary tree node.
 * function TreeNode(val, left, right) {
 *     this.val = (val===undefined ? 0 : val)
 *     this.left = (left===undefined ? null : left)
 *     this.right = (right===undefined ? null : right)
 * }
 */
/**
 * @param {TreeNode} root
 * @return {number[]}
 */
var preorderTraversal = function (root) {
    let ans = [];
    traverse(root,ans)
    return ans;
};
var traverse = function (root, ans) {
    if(!root) return null;
    ans.push(root.val);
    traverse(root.left,ans);
    traverse(root.right,ans);
}
```

迭代写法：

```
/**
 * Definition for a binary tree node.
 * function TreeNode(val, left, right) {
 *     this.val = (val===undefined ? 0 : val)
 *     this.left = (left===undefined ? null : left)
 *     this.right = (right===undefined ? null : right)
 * }
 */
/**
 * @param {TreeNode} root
 * @return {number[]}
 */
// 迭代
var preorderTraversal = (root) => {
    const list = [];
    const stack = [];

    // 当根节点不为空的时候，将根节点入栈
    if(root) stack.push(root)
    while(stack.length > 0) {
        const curNode = stack.pop()
        // 第一步的时候，先访问的是根节点
        list.push(curNode.val)

        // 我们先打印左子树，然后右子树
        // 所以先加入栈的是右子树，然后左子树
        if(curNode.right !== null) {
            stack.push(curNode.right)
        }
        if(curNode.left !== null) {
            stack.push(curNode.left)
        }
    }
    return list
}
```

## 589. N 叉树的前序遍历

同二叉树的前序遍历，输出根节点的值后从左到右递归的遍历子树。

```
递归写法：
/**
 * // Definition for a Node.
 * function Node(val, children) {
 *    this.val = val;
 *    this.children = children;
 * };
 */

/**
 * @param {Node} root
 * @return {number[]}
 */
var preorder = function (root) {
    let ans = [];
    traverse(root, ans);
    return ans;
};
var traverse = function (root, ans) {
    if (!root) return null;
    ans.push(root.val);
    for (x of root.children) {
        traverse(x,ans);
    }
    return ans;
}
```

迭代写法：

```
/**
 * // Definition for a Node.
 * function Node(val, children) {
 *    this.val = val;
 *    this.children = children;
 * };
 */

/**
 * @param {Node} root
 * @return {number[]}
 */
 /**解法二
 * @param {Node} root
 * @return {number[]}
 */
var preorder = function(root) {
    if (root === null) {
        return []
    }
    let array = []
    let stack = [root]
    while (stack.length) {
        let len = stack.length
        let node = stack.shift() // 弹出栈中第一个，先进先出
        array.push(node.val)
        if (node.children.length > 0) {
            stack = node.children.concat(stack) // 这里有别于层序遍历，用
node.children 连接 queue，而不是 queue.concat(node.children)这样就实现了前序遍历的效果
        }
    }
    return array
};
```

## 226. 翻转二叉树

递归的交换当前节点的左右子树即可。

```
递归写法:
/**
 * Definition for a binary tree node.
 * function TreeNode(val, left, right) {
 *     this.val = (val===undefined ? 0 : val)
 *     this.left = (left===undefined ? null : left)
 *     this.right = (right===undefined ? null : right)
 * }
 */
/**
 * @param {TreeNode} root
 * @return {TreeNode}
 */
var invertTree = function (root) {
    if (!root) return null;
    [root.left, root.right] = [root.right, root.left];
    invertTree(root.left);
    invertTree(root.right);
    return root;
};
```

```
迭代写法:
/**
 * Definition for a binary tree node.
 * function TreeNode(val, left, right) {
 *     this.val = (val===undefined ? 0 : val)
 *     this.left = (left===undefined ? null : left)
 *     this.right = (right===undefined ? null : right)
 * }
 */
/**
 * @param {TreeNode} root
 * @return {TreeNode}
 */
const invertTree = (root) => {
  if (!root) {
    return null;
  }
  let nowRoot = [root];

  while (nowRoot.length) {
    const nextRoot = [];
    for (let i = 0; i < nowRoot.length; i++) {
      if (nowRoot[i].left || nowRoot[i].right) {
        [nowRoot[i].left, nowRoot[i].right] = [nowRoot[i].right,
nowRoot[i].left];
        if (nowRoot[i].left) {
          nextRoot.push(nowRoot[i].left);
        }
        if (nowRoot[i].right) {
          nextRoot.push(nowRoot[i].right);
        }
      }
    }
    nowRoot = nextRoot;
  }
  return root;
};
```

## 剑指 Offer 32 - II. 从上到下打印二叉树 II

前序遍历整个二叉树，然后用一个二维数组来记录每层，存储每层的节点的值。

```
/**
 * Definition for a binary tree node.
 * function TreeNode(val) {
 *     this.val = val;
 *     this.left = this.right = null;
 * }
 */
/**
 * @param {TreeNode} root
 * @return {number[][]}
 */
var levelOrder = function(root) {
     let ans = new Array();
    getResult(root,0,ans);
    return ans;
};
var getResult = function(root,k,ans){
    if(!root) return null;
    if(k==ans.length) ans.push(new Array());
    ans[k].push(root.val);
    getResult(root.left,k+1,ans);
    getResult(root.right,k+1,ans);
}
```

迭代写法：，本题使用迭代有些麻烦，强烈建以递归

## 107. 二叉树的层序遍历 II

只需要将从上到下打印二叉树 II的结果进行反转即可。

```
/**
 * Definition for a binary tree node.
 * function TreeNode(val, left, right) {
 *     this.val = (val===undefined ? 0 : val)
 *     this.left = (left===undefined ? null : left)
 *     this.right = (right===undefined ? null : right)
 * }
 */
/**
 * @param {TreeNode} root
 * @return {number[][]}
 */
var levelOrderBottom = function (root) {
    let ans = new Array();
    getResult(root, 0, ans);
    for (let i = 0, j = ans.length - 1; i < j; i++, j--) {
        [ans[i], ans[j]] = [ans[j], ans[i]];
    }
    return ans;
};
var getResult = function (root, k, ans) {
    if (!root) return null;
    if (k == ans.length) ans.push(new Array());
    ans[k].push(root.val);
    getResult(root.left, k + 1, ans);
    getResult(root.right, k + 1, ans);
}
```

## 103. 二叉树的锯齿形层序遍历

只需要将从上到下打印二叉树 II的结果里面索引为奇数的数组中的元素反转即可。

```
/**
 * Definition for a binary tree node.
 * function TreeNode(val, left, right) {
```

```
 *     this.val = (val===undefined ? 0 : val)
 *     this.left = (left===undefined ? null : left)
 *     this.right = (right===undefined ? null : right)
 * }
 */
/**
 * @param {TreeNode} root
 * @return {number[][]}
 */
var zigzagLevelOrder = function (root) {
    let ans = new Array();
    getResult(root, 0, ans);
    for (let k = 1; k < ans.length; k += 2) {
        for (let i = 0, j = ans[k].length - 1; i < j; i++, j--) {
            [ans[k][i], ans[k][j]] = [ans[k][j], ans[k][i]];
        }
    }
    return ans;
};
var getResult = function (root, k, ans) {
    if (!root) return null;
    if (k == ans.length) ans.push(new Array());
    ans[k].push(root.val);
    getResult(root.left, k + 1, ans);
    getResult(root.right, k + 1, ans);
}
```

## 110. 平衡二叉树

先获取二叉树的高度，左子树的高度，右子树的高度，如果两者的差值的绝对值大于1那么就是非平衡二叉树，就返回

```
/**
 * Definition for a binary tree node.
 * function TreeNode(val, left, right) {
 *     this.val = (val===undefined ? 0 : val)
 *     this.left = (left===undefined ? null : left)
 *     this.right = (right===undefined ? null : right)
 * }
 */
/**
 * @param {TreeNode} root
 * @return {boolean}
 */
```

```
var isBalanced = function (root) {
    return getHeight(root) >= 0;
};
var getHeight = function (root) {
    if (!root) return 0;
    let l = getHeight(root.left);
    let r = getHeight(root.right);
    if (l < 0 || r < 0) return -2;
    if (Math.abs(l - r) > 1) return -2;
    return Math.max(l, r) + 1;
}
```