二叉树的基础遍历

二叉树的基础遍历分为：深度优先搜索 & 广度优先搜索

深度优先搜索

二叉树的递归遍历

二叉树的递归法遍历，通常用在深度优先搜索中

深度优先搜索：分为前序、中序、后序，前中后是依据**中结点**来分的

前序遍历：中左右

中序遍历：左右中

后序遍历：左右中

递归三要素：

（1）确定递归函数的参数和返回值的类型

（2）确定递归的终止条件

（3）确定单层递归的逻辑

题目：

144. 二叉树的前序遍历（简单）

145. 二叉树的后序遍历（简单）

94. 二叉树的中序遍历（简单）

二叉树的迭代遍历

迭代遍历的数据结构：栈

迭代遍历两大步骤：

（1）处理：把节点的值加入输出结果

（2）访问：遍历节点

前序遍历：前序遍历的顺序是中左右，可以让处理顺序和访问顺序一致。对处理的节点，将其出栈并加入输出结果后判断：如果它的右/左节点不为空，则访问右/左节点将其入栈（**先右后左**，符合出栈顺序）。循环条件是stack不为空

中序遍历：中序遍历的顺序是左中右，处理和访问顺序不同，需要借助指针cur来处理。如果cur不为空则把节点入栈，并让cur指向左节点；否则cur指回栈顶处理节点，将节点出栈并加入输出结果，并让cur指向右节点遍历。**循环条件是cur不为空或stack不为空**（否则可能输出不了完整的二叉树）

后序遍历：后序遍历的顺序和前序遍历原理类似。后序遍历可以转变成中左右→中右左→左右中的方式。中左右→中右左就将访问变成**先左后右**，中右左→左右中就将结果数组反转**（Collections.reverse(result);）**

*注意：需要单独判断二叉树为空的情况，直接返回结果数组

题目：

144. 二叉树的前序遍历（简单）

145. 二叉树的后序遍历（简单）

94. 二叉树的中序遍历（简单）

广度优先搜索

二叉树的层序遍历

层序遍历的数据结构：队列（**采用迭代法**）

基本程序框架：声明队列和结果变量，将root节点加入队列；设置**两个while循环**：**当队列不为空时**先记录当前队列大小n，**当n大于0时**将队首节点的左节点和右节点加入队列，然后把该队首节点弹出队列并使n减1，根据题目要求把节点值加入答案即可

二叉树的层序遍历：按照基本程序框架的步骤写，需要在while循环中设置结果数组的子数组，存储每一层的元素值

！！！“二叉树的层序遍历”是这类题的基础，其他题目几乎都是在它基础上进行细微改动

二叉树的层序遍历II：在二叉树的层序遍历基础上，反转最后的结果数组**（用Collections.reverse）**

二叉树的右视图：在二叉树的层序遍历基础上，当n为1时，把节点值加入结果数组中

二叉树的层平均值：在二叉树的层序遍历基础上，设置**double类型**的sum和num变量计算每层平均值（n=0时计算）

N叉树的层序遍历：用**for(Node child: queue.peek().children)**访问每个节点的孩子节点，其它和二叉树的层序遍历完全一致

在每个树行中找到最大值：在二叉树的层序遍历基础上，增加max变量（初始化为无穷小），在每一层循环内不断得到该层的最大值

填充每个节点的下一个右侧节点指针：需要设置cur、next指针，初始指向队列头节点（queue.poll()）。大while循环对cur指针操作（加入节点），小while循环对next指针操作。每一层都是在加入下一层节点的时候进行填充指针的操作：具体为把next赋给cur.next，并让cur等于next。注意填充指针的操作次数等于该层节点数-1，因此小循环判断条件应当为--n>0。最后返回root即可得到答案

二叉树的最大深度：在二叉树的层序遍历基础上，增加depth变量（初始为0），**只要队列不为空（即进入栈不为空的循环）就把depth加1**

二叉树的最小深度：如果遍历到某层某节点既无左节点也无右节点，直接返回depth结束程序，其余过程和二叉树的最大深度相同

题目：

102. 二叉树的层序遍历（中等）

107. 二叉树的层序遍历II（中等）

199. 二叉树的右视图（中等）

637. 二叉树的层平均值（简单）

429. N叉树的层序遍历（中等）

515. 在每个树行中找到最大值（中等）

116. 填充每个节点的下一个右侧节点指针（中等）

117. 填充每个节点的下一个右侧节点指针II（中等，和上题一模一样）

104. 二叉树的最大深度（简单）

111. 二叉树的最小深度（简单）