拉勾教育
—互联网人实战大学—

二叉树+BFS+DFS：二叉树最大深度

简单/二叉树、BFS、DFS

学习目标

拉勾教育
—互联网人实战大学—

- 掌握二叉树的特点
- 掌握树的BFS(广度优先遍历)思想
- 掌握树的DFS(深度优先遍历)思想
- 掌握并熟练使用二叉树的BFS + DFS

题目描述

给定一个二叉树，找出其最大深度。

二叉树的深度为根节点到最远叶子节点最长路径上的节点数

说明：叶子节点是指没有子节点的节点

示例：

给定二叉树 `[3,9,20,null,null,15,7]`，

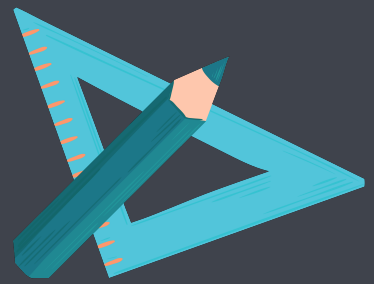
```
    3
   / \
  9  20
    /  \
   15   7
```

返回它的最大深度 3 。

一、Comprehend 理解题意

题目主干：给定一个二叉树，找出其最大深度

- 二叉树
  - 树的概念
  - 二叉树的概念
- 最大深度
  - 深度的概念



一、Comprehend 理解题意

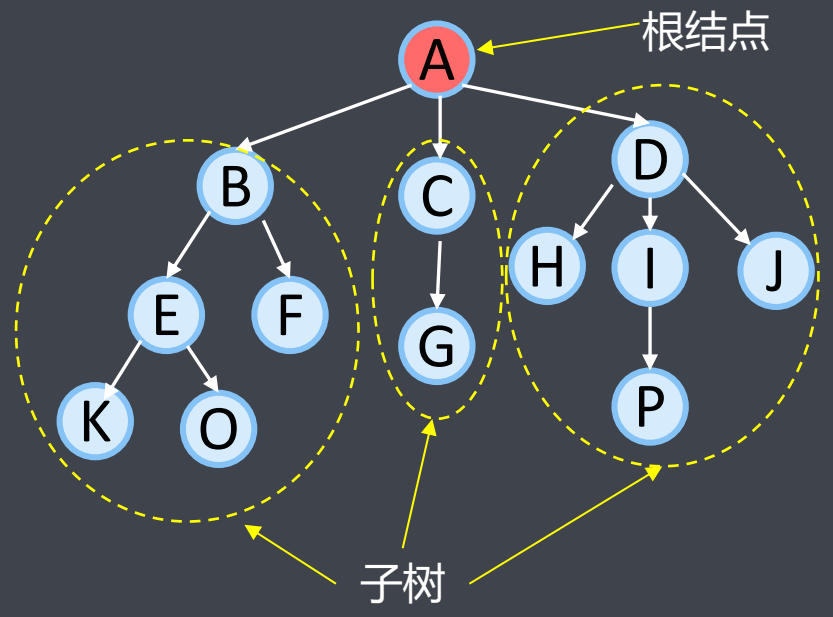
拉勾教育
—互联网人实战大学—

树的定义：

树(tree)是n(n>0)个结点(node)的有限集T，其中：

- 有且仅有一个特定的结点，称为根(root)
- 当n>1时，其余结点可分为m(m>0)个互不相交的有限集，每个集合本身是一棵树，称为根的子树(subtree)

只有根结点的树



一、Comprehend 理解题意

树的性质：

- 结点(node)：树的元素，包括数据项+若干指向其子树的分支
- 结点的度(degree)：结点拥有的子树的数目
- 叶子(leaf)：度为0的结点
- 结点的层次(level)：从根结点算起，根为第一层，其孩子为第二层 ……

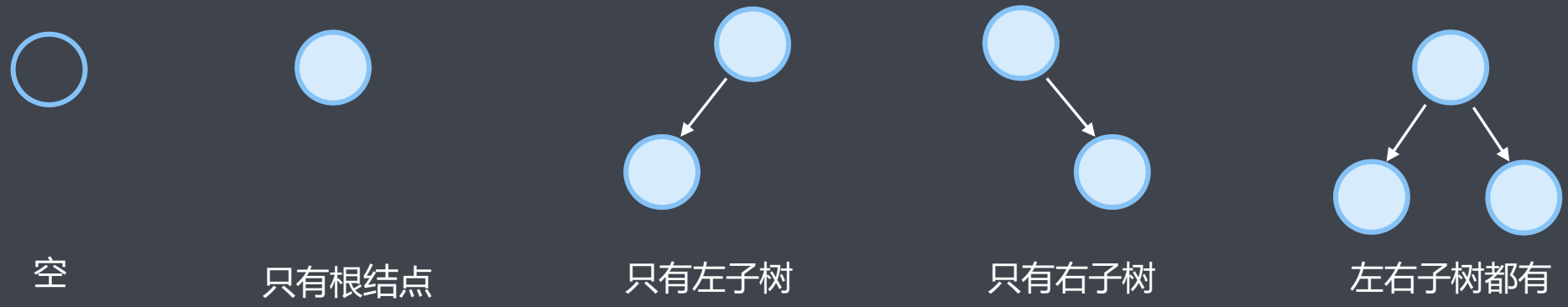
一、Comprehend 理解题意

二叉树的定义：

二叉树是n(n>=0)个结点的有限集，它或为空树，或由两棵分别称为左子树和右子树的互不相交的二叉树构成

二叉树的性质:

1. 每个结点最多有两棵子树
2. 二叉树的左右子树次序不能颠倒



二叉树的五种形态

一、Comprehend 理解题意

二叉树的代码实现：

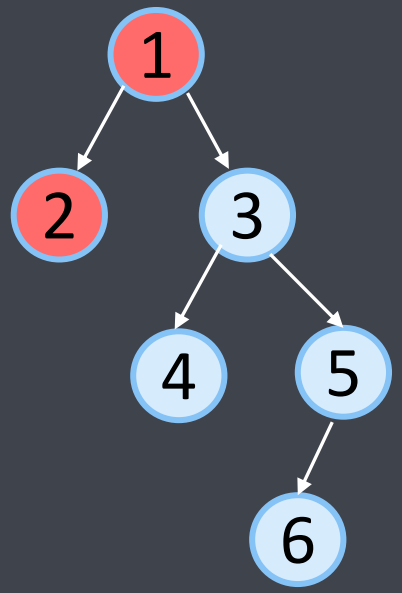

```
// 定义一棵二叉树
public class TreeNode {
    // 定义节点的值
    int val;
    // 定义左子树
    TreeNode left;
    // 定义右子树
    TreeNode right;
    // 定义初始化方法
    TreeNode(int x) { val = x; }
}
```

一、Comprehend 理解题意

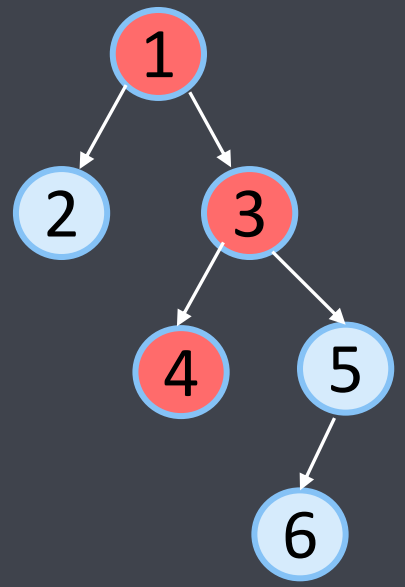
拉勾教育
—互联网人实战大学—

细节问题：

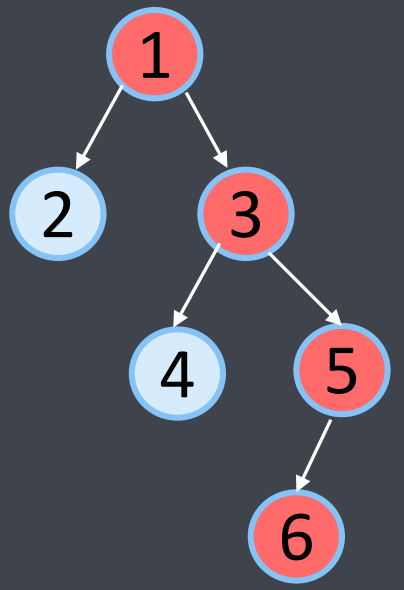
二叉树最大深度：该二叉树所有深度的最大值



深度=2



深度=3

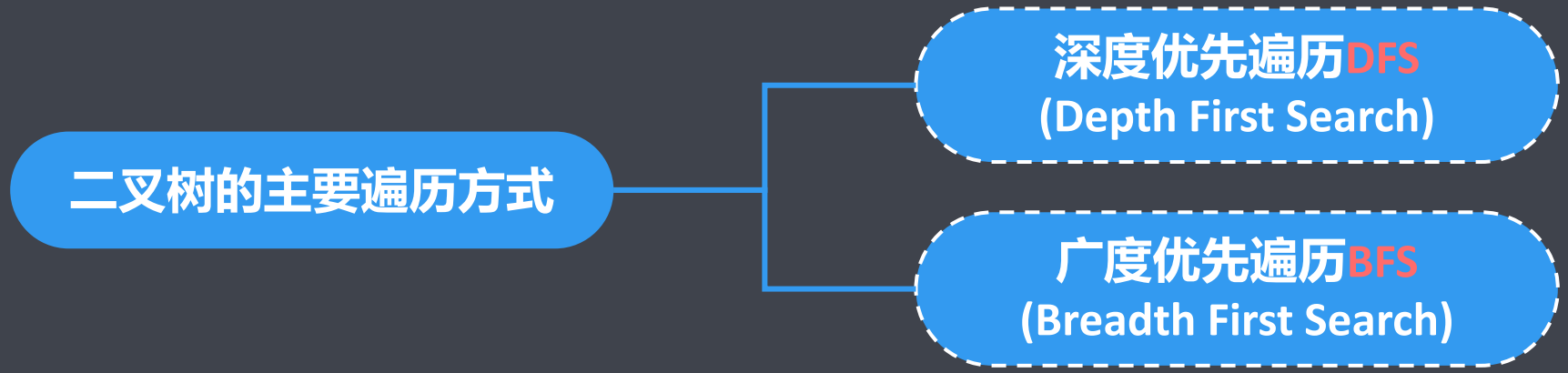


深度=4

最大深度=4

二. Choose 数据结构及算法思维选择

在本题中，若想求得二叉树最大深度，需求出二叉树所有的深度，那么就需要遍历树的所有结点。



方案一：广度优先遍历

广度优先遍历所有结点，找出最大深度

- 数据结构：树，队列
- 算法思维：广度优先

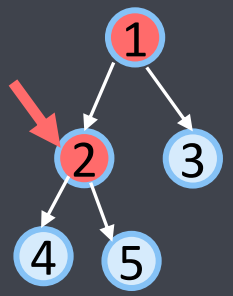
方案二：深度优先遍历（优化解法）

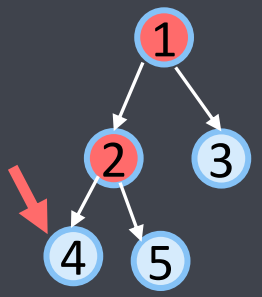
深度优先遍历所有结点，找出最大深度

- 数据结构：树
- 算法思维：深度优先

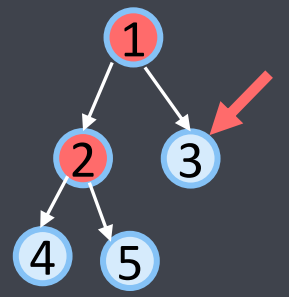
二. Choose 数据结构及算法思维选择

关键知识点：DFS+BFS





选择一



选择二

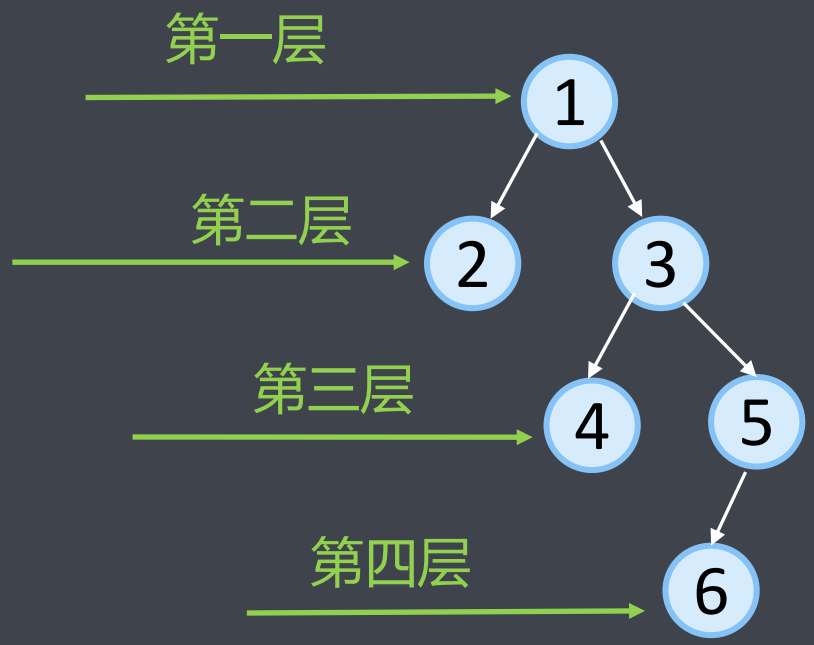
在遍历树的时候，总是从一个结点出发，遍历其子结点。在上图遍历到结点2后，有两种遍历选择：

1. **先遍历下一层**结点（结点4）及后面所有层直至不能往下**再回退**
2. **先回退**将根结点剩余子结点（结点3）遍历完后**再遍历下一层**结点

第一种遍历方式称为**DFS**(深度优先遍历)，第二种遍历方式称为**BFS**(广度优先遍历)

三、Code 基本解法及编码实现

解法一：BFS



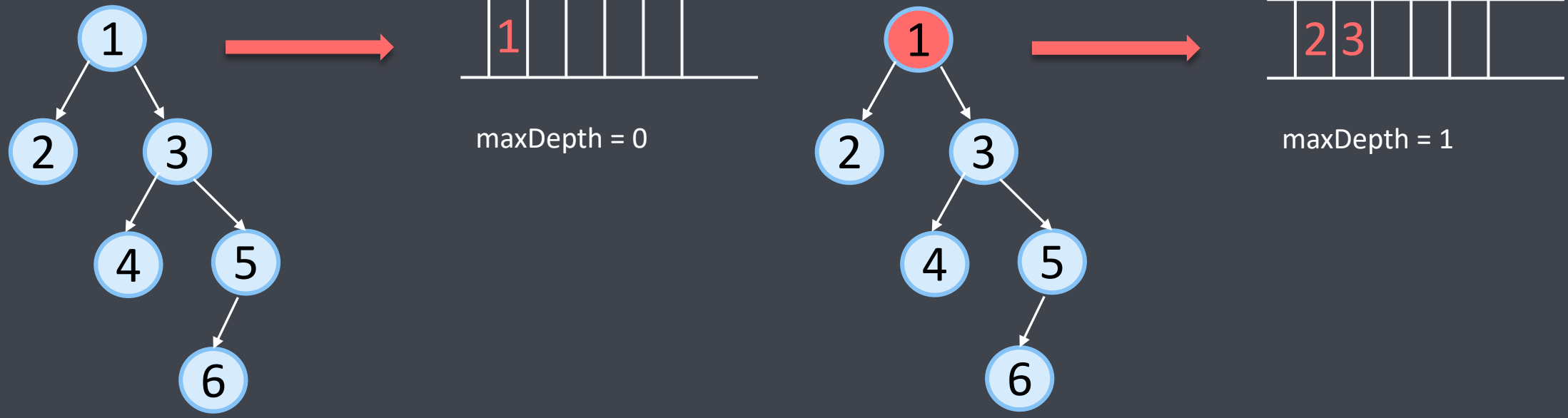
思想：自上而下，将同一层结点遍历完后才开始遍历下一层结点

思考：

- 是否需要处理边界问题？
- 在遍历根结点（结点1）时如何记录其下一层节点（结点2，结点3）？
- 如何确定某一层结点已被遍历完？
- 在BFS过程中如何更新我们的目标值（二叉树最大深度）？

三、Code 基本解法及编码实现

BFS算法思路剖析

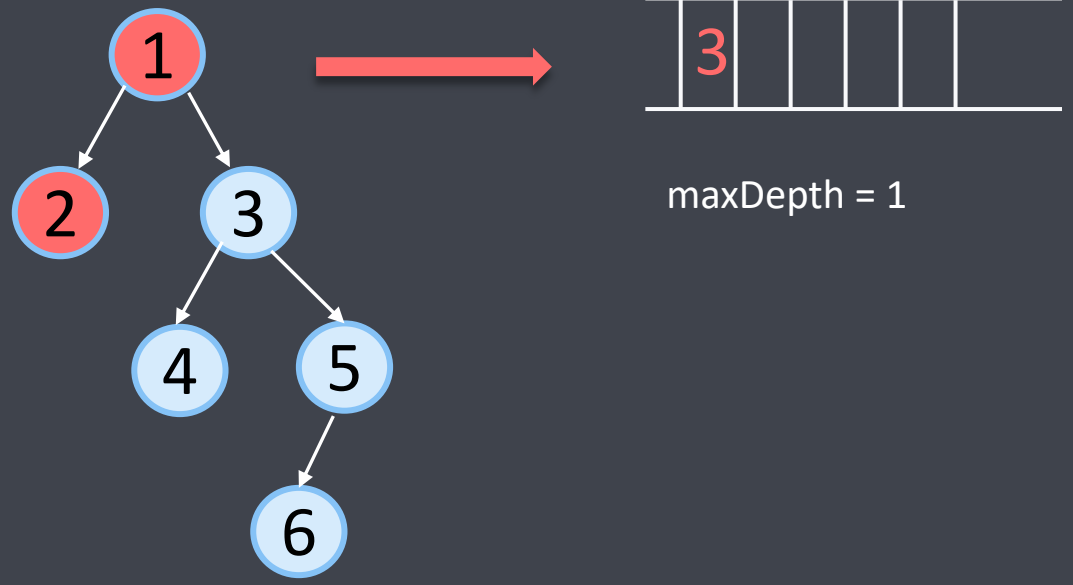


1 将根结点压入队列

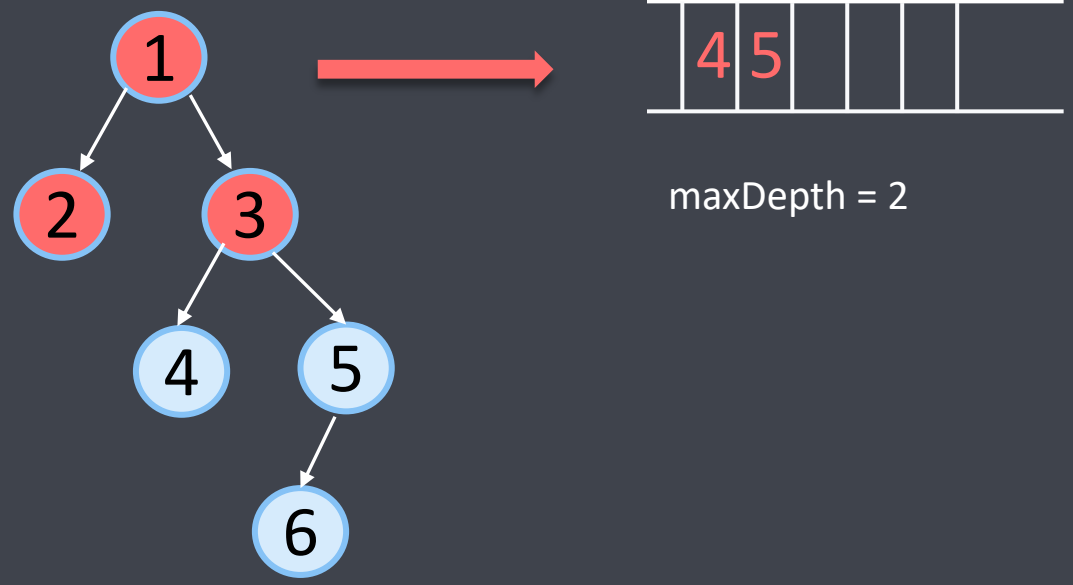
2 第一层结点出队列，其子结点入队列，更新目标值

三、Code 基本解法及编码实现

BFS算法思路剖析



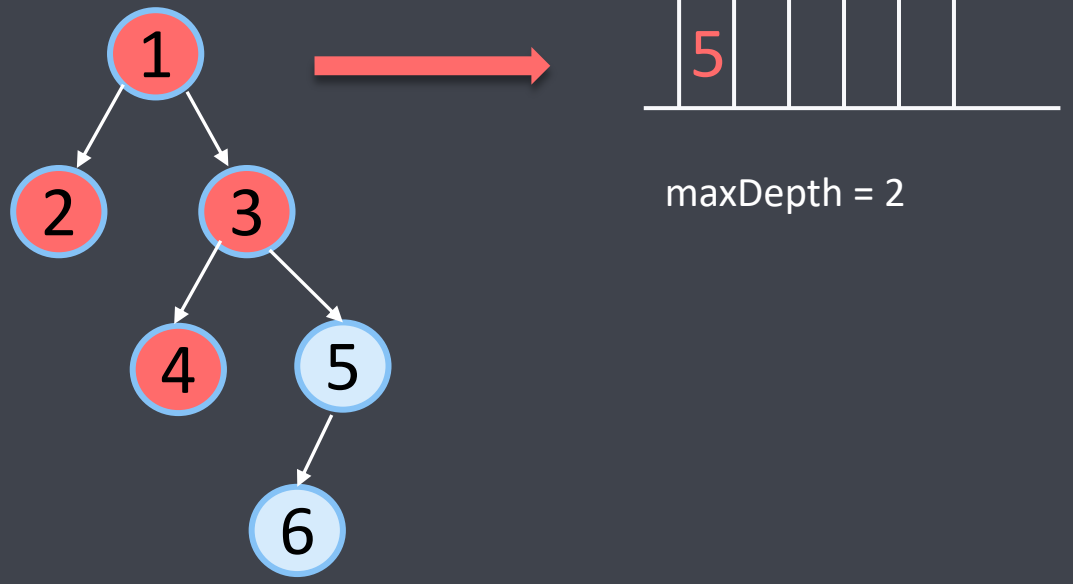
3 遍历结点2，注意此时无需更新目标值



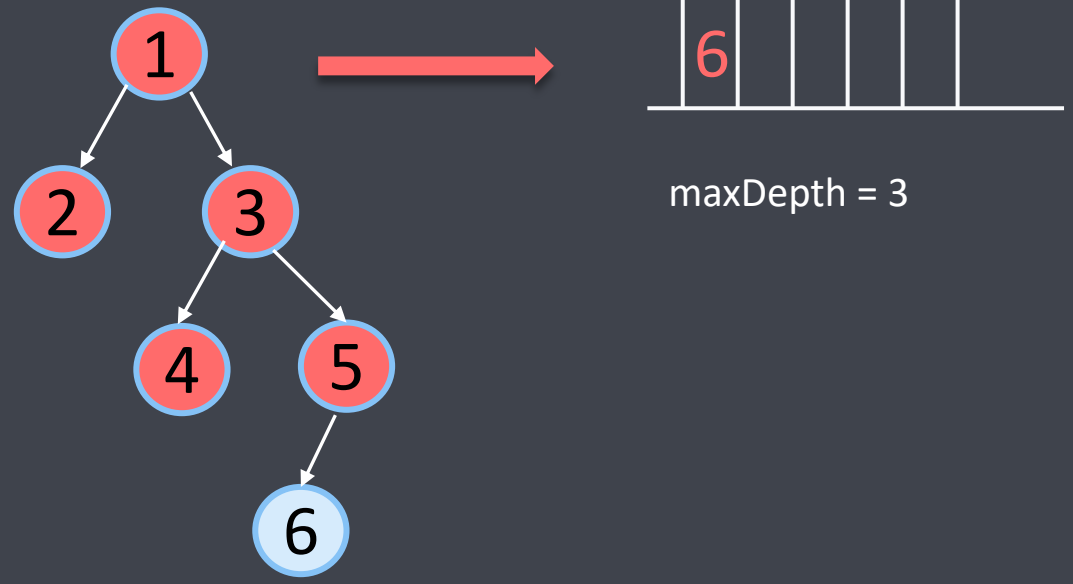
4 遍历结点3，第二层结点遍历完毕，更新目标值

三、Code 基本解法及编码实现

BFS算法思路剖析



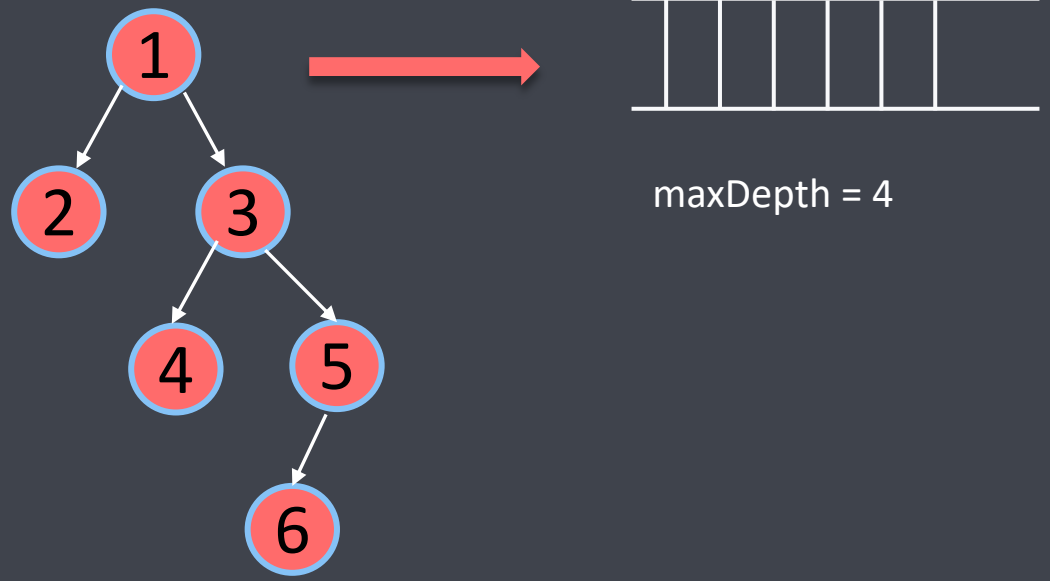
5 遍历结点4，注意此时无需更新目标值



6 遍历节点6，第三层节结点遍历完，更新目标值

三、Code 基本解法及编码实现

BFS算法思路剖析



6 遍历节结点6，第四层节点遍历完，更新目标值

三、Code 基本解法及编码实现

解法一：BFS

思考：

- 是否需要处理边界问题？
  - 处理树为空的情况
- 在遍历根结点（结点1）时如何记录其下一层节点（结点2，结点3）？
  - 遍历当前结点时，将其左右子结点加入队列
- 如何确定某一层结点已被遍历完？
  - 在更新目标值后，队列中的结点就是下一层的全部结点，记录此时队列的大小
- 在BFS过程中如何更新我们的目标值（二叉树最大深度）？
  - 遍历完一层结点后，目标值+1

三、Code 基本解法及编码实现

解法一：BFS复杂度分析

时间复杂度：O(n)

- n是二叉树节点数目
- 每个节点会被遍历一次

空间复杂度：

- 取决于队列需要存储的元素数量，最差可以达到O(n)

三、Code 基本解法及编码实现

解法一：BFS参考代码

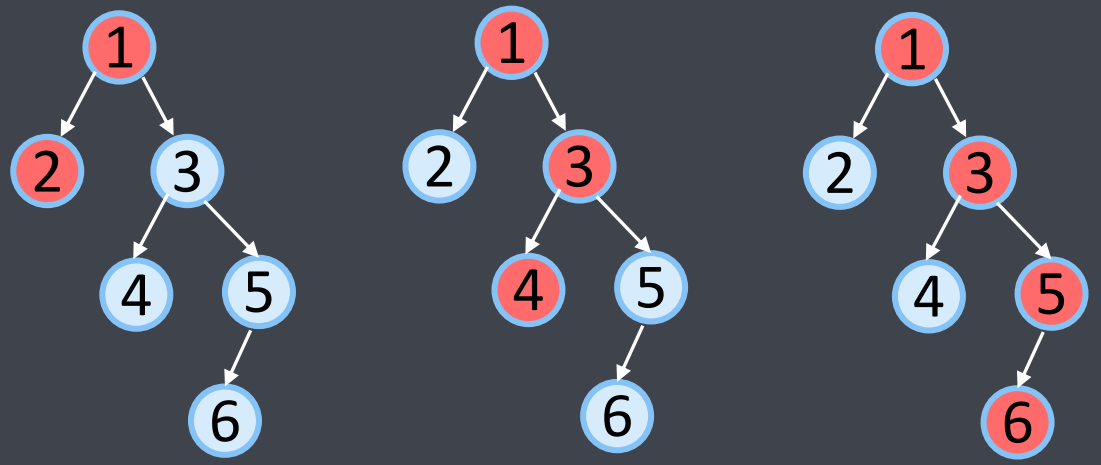
Java编码实现

```java
class Solution {
    public int maxDepth(TreeNode root) {
        if (root == null) {return 0;} // 考虑树为空的特殊情况 BFS无法自动处理
        Queue<TreeNode> queue = new LinkedList<TreeNode>();// 使用队列来记录各层节点
        queue.offer(root);// 根节点入队
        int res = 0;// 目标值
        while (!queue.isEmpty()) { // 判断是否还有没有遍历完的节点
            int size = queue.size(); // 开始遍历新一层节点前，队列里即为新一层全部节点
            while (size > 0) {  // 需将这一层节点全都遍历完
                TreeNode node = queue.poll();// 遍历节点
                if (node.left != null) {
                    queue.offer(node.left);// 左子树入队列
                }
                if (node.right != null) {
                    queue.offer(node.right);// 右子树入队列
                }
                size--;
            }
            res++;// 新一层节点遍历完成，目标值 +1
        }
        return res;
    }
}
```

四. Consider 思考更优解

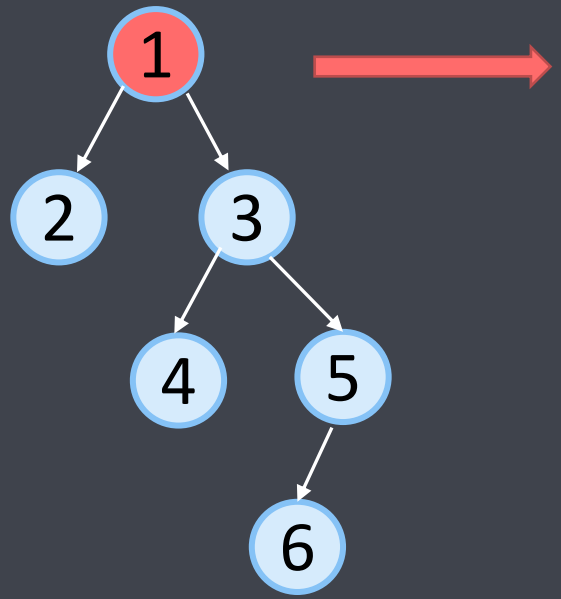
解法二：DFS（递归）（优化解法）

- BFS是横向遍历每一个节点，需要O（n）的空间复杂度
- DFS将会纵向遍历节点，在遍历每个节点时，即可得到当前节点至根节点的深度，无需存储



五. Code 最优解思路及编码实现

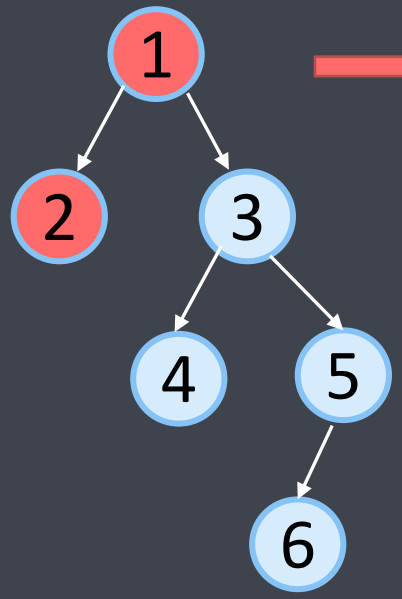
DFS算法思路剖析



step 1

Height

H= 1+ Max(H(2), H(3))



step 2

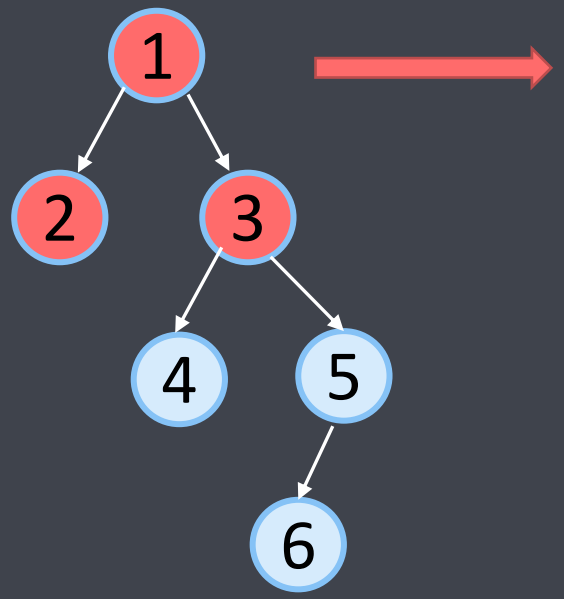
Height

H= 1+ Max(H(2), H(3))

H(2)=1+Max(0,0)=1

五. Code 最优解思路及编码实现

DFS算法思路剖析



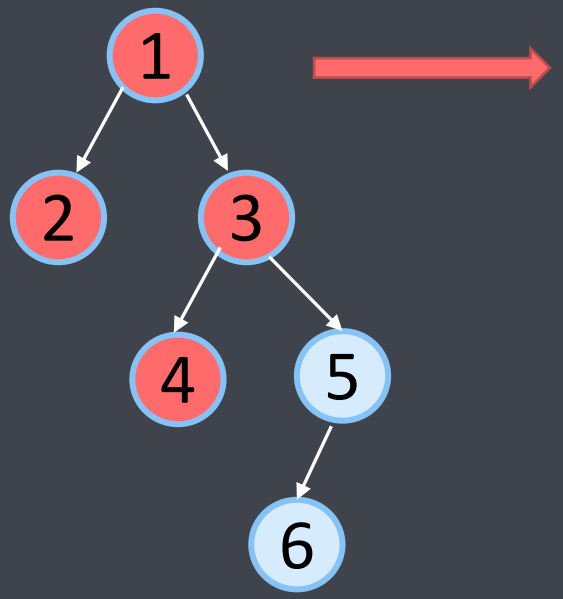
step 3

Height

H= 1+ Max(H(2), H(3))

H(2)=1+Max(0,0)=1

H(3)=1+Max(H(4),H(5))



step 4

Height

H= 1+ Max(H(2), H(3))

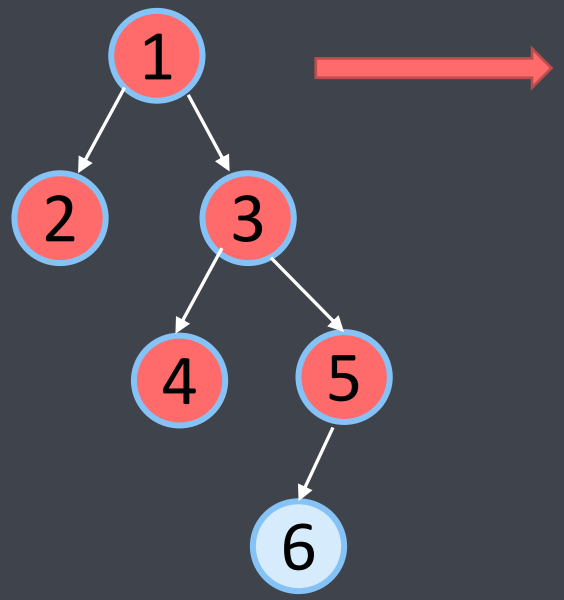
H(2)=1+Max(0,0)=1

H(3)=1+Max(H(4),H(5))

H(4)=1+Max(0,0)=1

五. Code 最优解思路及编码实现

DFS算法思路剖析



step 5

Height

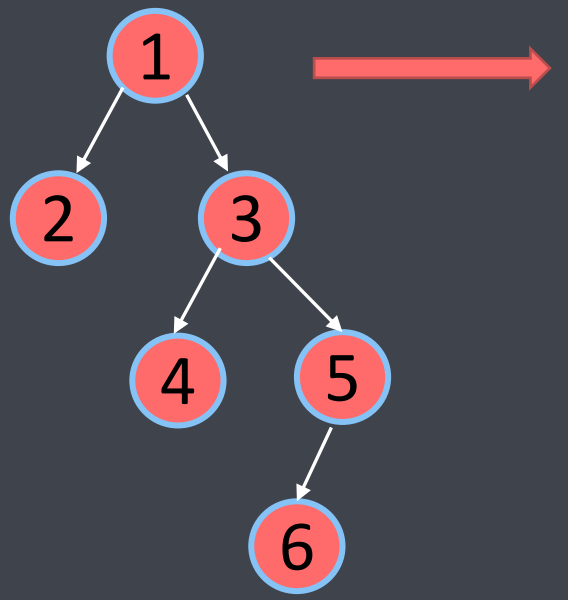
H= 1+ Max(H(2), H(3))

H(2)=1+Max(0,0)=1

H(3)=1+Max(H(4),H(5))

H(4)=1+Max(0,0)=1

H(5)=1+Max(H(6),0)



step 6

Height

H= 1+ Max(H(2), H(3))

H(2)=1+Max(0,0)=1

H(3)=1+Max(H(4),H(5))

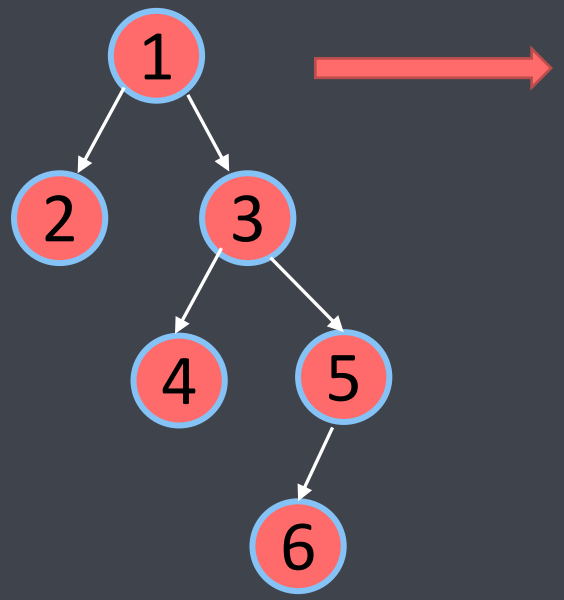
H(4)=1+Max(0,0)=1

H(5)=1+Max(H(6),0)

H(6)=1+Max(0,0)=1

五. Code 最优解思路及编码实现

DFS算法思路剖析



step 7

Height

H= 1+ Max(H(2), H(3))

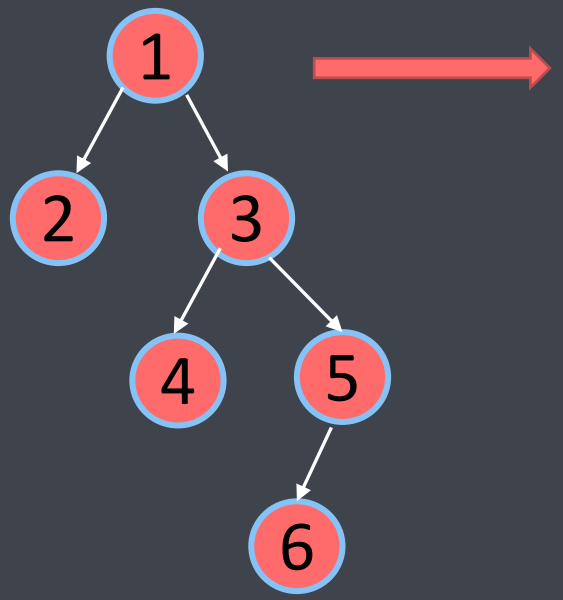
H(2)=1+Max(0,0)=1

H(3)=1+Max(H(4),H(5))

H(4)=1+Max(0,0)=1

H(5)=1+Max(H(6),0)=2

H(6)=1+Max(0,0)=1



step 8

Height

H= 1+ Max(H(2), H(3))

H(2)=1+Max(0,0)=1

H(3)=1+Max(H(4),H(5))=3

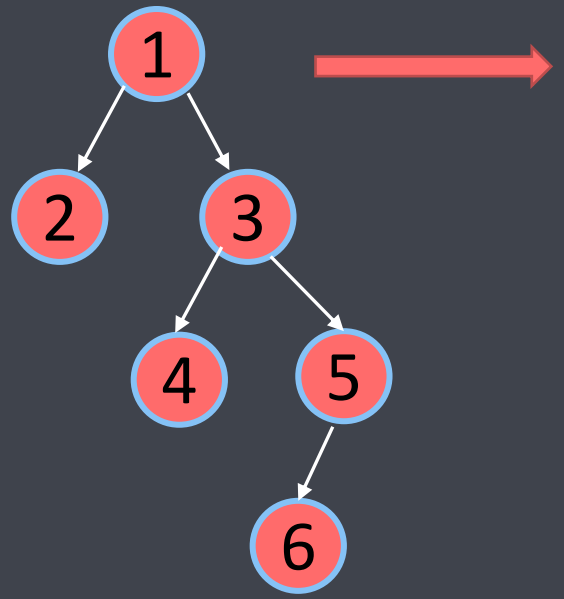
H(4)=1+Max(0,0)=1

H(5)=1+Max(H(6),0)=2

H(6)=1+Max(0,0)=1

五. Code 最优解思路及编码实现

DFS算法思路剖析



step 9

Height

H= 1+ Max(H(2), H(3))=4

H(2)=1+Max(0,0)=1

H(3)=1+Max(H(4),H(5))=3

H(4)=1+Max(0,0)=1

H(5)=1+Max(H(6),0)=2

H(6)=1+Max(0,0)=1

五. Code 最优解思路及编码实现

解法二：DFS复杂度分析

时间复杂度：O(n)

- n是二叉树节点数目
- 每个节点会被遍历一次

空间复杂度：O(h)

- h是二叉树的高度
- 递归需要栈，而栈的深度取决于二叉树的高度

五. Code 最优解思路及编码实现

解法二：深度优先遍历（递归）参考代码

Java编码实现

```
// 递归三要素
// 1）确定函数等价关系式（参数，返回值）。参数是传入树的根节点，返回值是树的深度
public int maxDepth(TreeNode root) {
    // 2)确定结束条件。如果节点为空，返回0
    // 且此处可以处理二叉树为空的corner case
    if(root == null) { return 0; }
    // 3)函数主功能。分别求左右子树最大深度，返回左右子树深度最大值 + 1
    //   即为当前节点为根节点的树的最大深度
    int leftTreeDepth=maxDepth(root.left);
    int rightTreeDepth=maxDepth(root.right);
    return Math.max(leftTreeDepth,rightTreeDepth)+1
}
```

六. Change 变形延伸

拉勾教育
—互联网人实战大学—

题目变形

- （练习）二叉树的层次遍历(Leetcode 104)

延伸扩展

- 在树的问题中使用DFS（递归）往往可以使复杂问题迎刃而解
- 树这种数据结构在数据库中有广泛的作用，可以提高查询速度
- 面试考察中重点考察的数据结构

本题来源

- Leetcode 104 https://leetcode-cn.com/problems/maximum-depth-of-binary-tree/

总结

- 掌握二叉树的特点
- 掌握树的BFS(广度优先遍历)思想
- 掌握树的DFS(深度优先遍历)思想
- 掌握并熟练使用二叉树树的BFS + DFS

课后练习

1. 二叉树的镜像(剑指offer 27 /简单)
2. N叉树的最大深度（ Leetcode 559 /简单 ）
3. 二叉树的右视图（ Leetcode 199 /中等 ）
4. 二叉树中最大路径和（ Leetcode 124 /困难 ）

课后练习

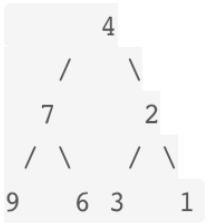
拉勾教育
—互联网人实战大学—

1. 二叉树的镜像(剑指offer27/简单)

提示：本题中的镜像是指左右对称镜像。

例如输入：

镜像输出：



示例 1：

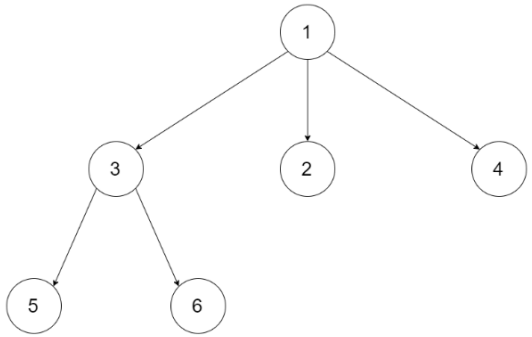
```
输入：root = [4,2,7,1,3,6,9]
输出：[4,7,2,9,6,3,1]
```

课后练习

2. N叉树的最大深度（Leetcode 559/简单）

提示：N叉树与二叉树的区别在于每个根节点可以有多于2个的子节点。

例如，给定一个 3叉树 ：



我们应返回其最大深度，3。

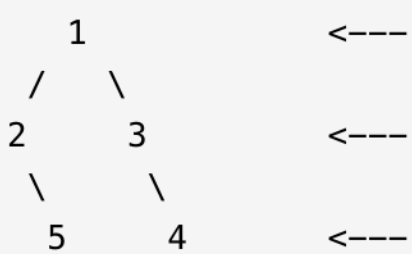
课后练习

拉勾教育
—互联网人实战大学—

3. 二叉树的右视图（Leetcode 199 /简单）

提示：想象你站在一个二叉树的右侧，按照从顶部到底部的顺序，返回从右侧看到的所有节点

输入： [1,2,3,null,5,null,4]
输出： [1, 3, 4]
解释：



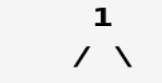
课后练习

拉勾教育
—互联网人实战大学—

4. 二叉树中最大路径和（Leetcode 124/困难）

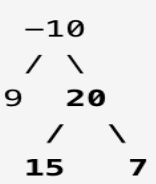
提示：本题中二叉树是非空二叉树，路径被定义为一条从树中任意节点出发，沿父节点-子节点连接，达到任意节点的序列。该路径至少包含一个节点，且不一定经过根节点。

输入： [1,2,3]



输出： 6

输入： [-10,9,20,null,null,15,7]



输出： 42

拉勾教育

—互联网人实战大学—

下载「拉勾教育App」
获取更多内容