[**104. 二叉树的最大深度**](https://leetcode.cn/problems/maximum-depth-of-binary-tree/)

已解答

简单

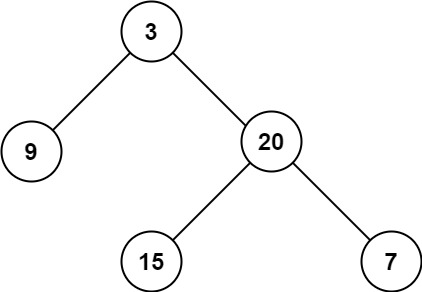
相关标签

相关企业

给定一个二叉树 root ，返回其最大深度。

二叉树的 **最大深度** 是指从根节点到最远叶子节点的最长路径上的节点数。

**示例 1：**



**输入：**root = [3,9,20,null,null,15,7]

**输出：**3

**示例 2：**

**输入：**root = [1,null,2]

**输出：**2

**提示：**

* 树中节点的数量在 [0, 104] 区间内。
* -100 <= Node.val <= 100

代码：

# Definition for a binary tree node.

# class TreeNode(object):

#     def \_\_init\_\_(self, val=0, left=None, right=None):

#         self.val = val

#         self.left = left

#         self.right = right

class Solution(object):

    def maxDepth(self, root):

        """

        :type root: TreeNode

        :rtype: int

        """

        if root is None:

            return 0

        else:

            left=self.maxDepth(root.left)

            right=self.maxDepth(root.right)

            return max(left,right)+1

[**144. 二叉树的前序遍历**](https://leetcode.cn/problems/binary-tree-preorder-traversal/)

已解答

简单

相关标签

相关企业

给你二叉树的根节点 root ，返回它节点值的 **前序** 遍历。

**示例 1：**



**输入：**root = [1,null,2,3]

**输出：**[1,2,3]

**示例 2：**

**输入：**root = []

**输出：**[]

**示例 3：**

**输入：**root = [1]

**输出：**[1]

**示例 4：**



**输入：**root = [1,2]

**输出：**[1,2]

**示例 5：**



**输入：**root = [1,null,2]

**输出：**[1,2]

**提示：**

* 树中节点数目在范围 [0, 100] 内
* -100 <= Node.val <= 100

代码：# Definition for a binary tree node.

# class TreeNode(object):

#     def \_\_init\_\_(self, val=0, left=None, right=None):

#         self.val = val

#         self.left = left

#         self.right = right

class Solution(object):

    def preorderTraversal(self, root):

        """

        :type root: TreeNode

        :rtype: List[int]

        """

        if root is None:

            return []

        return [root.val]+self.preorderTraversal(root.left)+self.preorderTraversal(root.right)

[**145. 二叉树的后序遍历**](https://leetcode.cn/problems/binary-tree-postorder-traversal/)

已解答

简单

相关标签

相关企业

给你一棵二叉树的根节点 root ，返回其节点值的 **后序遍历**。

**示例 1：**



**输入：**root = [1,null,2,3]

**输出：**[3,2,1]

**示例 2：**

**输入：**root = []

**输出：**[]

**示例 3：**

**输入：**root = [1]

**输出：**[1]

**提示：**

* 树中节点的数目在范围 [0, 100] 内
* -100 <= Node.val <= 100

代码:

# Definition for a binary tree node.

# class TreeNode(object):

#     def \_\_init\_\_(self, val=0, left=None, right=None):

#         self.val = val

#         self.left = left

#         self.right = right

class Solution(object):

    def postorderTraversal(self, root):

        """

        :type root: TreeNode

        :rtype: List[int]

        """

        if root is None:

            return []

        return self.postorderTraversal(root.left)+self.postorderTraversal(root.right)+[root.val]

[**217. 存在重复元素**](https://leetcode.cn/problems/contains-duplicate/)

已解答

简单

相关标签

相关企业

给你一个整数数组 nums 。如果任一值在数组中出现 **至少两次** ，返回 true ；如果数组中每个元素互不相同，返回 false 。

**示例 1：**

**输入：**nums = [1,2,3,1]

**输出：**true

**示例 2：**

**输入：**nums = [1,2,3,4]

**输出：**false

**示例 3：**

**输入：**nums = [1,1,1,3,3,4,3,2,4,2]

**输出：**true

**提示：**

* 1 <= nums.length <= 105
* -109 <= nums[i] <= 109

代码：

class Solution{

    public boolean containsDuplicate(int[] nums) {

        HashMap<Integer,Integer>h1=new HashMap<Integer,Integer>();

        for(int i=0;i<nums.length;i++) {

            if(h1.containsKey(nums[i])) {

                return true;

            }

            h1.put(nums[i],i);

        }

        return false;

    }

}

[**219. 存在重复元素 II**](https://leetcode.cn/problems/contains-duplicate-ii/)

已解答

简单

相关标签

相关企业

给你一个整数数组 nums 和一个整数 k ，判断数组中是否存在两个 **不同的索引** i 和 j ，满足 nums[i] == nums[j] 且 abs(i - j) <= k 。如果存在，返回 true ；否则，返回 false 。

**示例 1：**

**输入：**nums = [1,2,3,1], k = 3

**输出：**true

**示例 2：**

**输入：**nums = [1,0,1,1], k = 1

**输出：**true

**示例 3：**

**输入：**nums = [1,2,3,1,2,3], k = 2

**输出：**false

**提示：**

* 1 <= nums.length <= 105
* -109 <= nums[i] <= 109
* 0 <= k <= 105

代码：

class Solution {

    public boolean containsNearbyDuplicate(int[] nums, int k) {

        HashMap<Integer,Integer>h1=new HashMap<Integer,Integer>();

        for(int i=0;i<nums.length;i++) {

            if(h1.containsKey(nums[i])) {

                if(Math.abs(i-h1.get(nums[i]))<=k) {

                    return true;

                }

            }

            h1.put(nums[i],i);

        }

        return false;

    }

}