# 描述

给你一棵二叉树，请你返回层数最深的叶子节点的和。

**示例：**

输入：root = [1,2,3,4,5,null,6,7,null,null,null,null,8]

输出：15

**提示：**

树中节点数目在 1 到 10^4 之间。

每个节点的值在 1 到 100 之间。

# 分析

## 深度优先搜索

方法一：深度优先搜索

我们可以使用深度优先搜索的方法解决这个问题。

我们从根节点开始进行搜索，在搜索的同时记录当前节点的深度 dep。我们维护两个全局变量 maxdep 和total，其中 maxdep 表示搜索到的节点的最大深度，total 表示搜索到的深度等于 maxdep 的节点的权值之和。

当我们搜索到一个新的节点 x 时，会有以下三种情况：

节点x的深度dep小于maxdep，那么我们可以忽略节点x，继续进行搜索；

节点x的深度dep等于maxdep，那么我们将节点x的权值添加到total中；

节点x的深度dep大于maxdep，此时我们找到了一个深度更大的节点，因此需要将maxdep置为dep，并将total置为节点x的权值。

在深度优先搜索结束之后，深度最大的叶子节点的权值之和即存储在total 中。

## 广度优先搜索

方法二：广度优先搜索

我们同样可以使用广度优先搜索的方法解决这个问题。

除了搜索的顺序不同之外，实现的细节与深度优先搜索的方法完全相同。

# 代码

## C

## C++

## Java

## Python