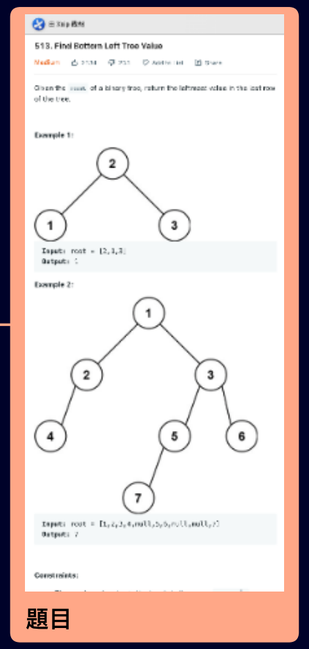
513. Find Bottom Left Tree Value



題目

解析

- 樹左下角的值
- 最底層最左邊節點的值
- “最底層”也就是二元樹的最大深度，最大深度上的那一層一定是最後一層。
- 求二元樹最大深度
  - 自頂向下
    - 前序遍歷
    - 從根節點遞迴到葉子節點
    - 對於每個節點來說，先判斷是否為葉子節點。
    - 如果不是葉子節點，因為要的是“最左邊”的節點，所以每次先遍歷左子樹，再遍歷右子樹
  - 自左向右
    - 層次遍歷
    - 自左向右，就是從根節點開始，一層一層的遍歷二元樹
    - 最後一層的第一個節點就是我們想要的值
  - 自底向上
    - 根據這道題的自身還有“最左邊”的特點 不考慮

題解

- 遞迴法
  - 自頂向下 即前序遍歷
    - 遞迴三部曲
      1. 確定遞迴函數的參數和返回值
         - 需要類裡的兩個全域變數，maxDepth用來記錄最大深度，res記錄最大深度最左節點的數值。
      2. 確定終止條件
         - if root == None:
return
      3. 確定單層遞迴的邏輯
         - 前序遍歷的順序是：根、左子樹、右子樹。
         - 本題同樣也是這個順序：**判斷節點、**遞迴左子樹，遞迴右子樹。
         - 這題中我們其實要維護兩個深度，一個是當前的遍歷到的最大深度 maxDepth，另一個是當前節點所處的深度 leftDepth。
         - 如果當前節點是葉子節點，且 leftDepth > maxDepth 的時候，更新 maxDepth 和當前的結果 res。
         - 這證明當前遍歷的節點是新的一層最先被遍歷的節點，因為我們優先進行的是左子樹的遍歷，所以它肯定是當前層最左邊的節點。

```
# 如果當前節點是葉子節點
if root.left == None and root.right == None:
  # 當前葉子節點的深度大於之前保存的最大深度
  # 證明是當前深度的最左邊葉子節點，因為先遞迴左子樹
  # 此時更新最大深度，更新結果值
  if leftDepth > self.maxDepth:
    self.maxDepth = leftDepth
    self.res = root.val

# 遞迴左子樹
self.leftValue(root.left, leftDepth + 1)
# 遞迴右子樹
self.leftValue(root.right, leftDepth + 1)
```