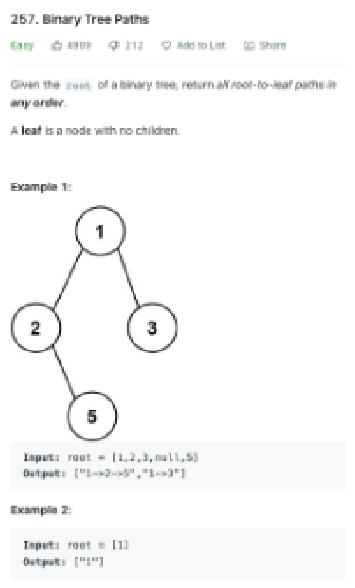
257. Binary Tree Paths

題目



257. Binary Tree Paths

Easy 4909 212 Add to List Share

Given the `root` of a binary tree, return *all root-to-leaf paths* in **any order**.

A **leaf** is a node with no children.

Example 1:

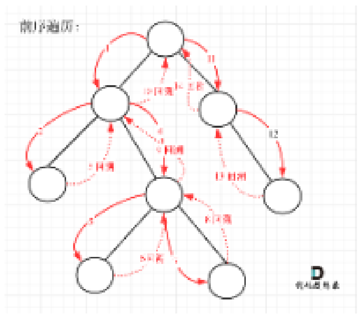
```
Input: root = [1,2,3,null,5]
Output: ["1->2->5","1->3"]
```

Example 2:

```
Input: root = [1]
Output: ["1"]
```

題目

遞歸

- 前序遍歷
  - 自頂向下的解法是從根節點遞迴到葉子節點，計算這一條路徑上的深度，並更新維護最大深度。每次先維護根節點的深度，再遞迴左子樹、右子樹。
  - 在本題的所有路徑中, 自頂而下的解法是從根節點遞歸到葉子結點
    只不過這次不是計算路徑上的深度，而是記錄路徑上的節點，並更新維護路徑。
    每次都是先從根節點開始，先遞迴左子樹，再遞迴右子樹。
  - 

    前序遍歷以及回溯過程圖

遞歸題解

- 遞歸三步驟
  - 1.遞迴函數函數參數以及返回值
    - 傳入根節點，記錄每一條路徑的path，和存放結果集的result，這裡遞迴不需要返回值
    - 建立getPaths 傳入root(根節點), path(路徑), res(結果集)
  - 2.確定遞歸的終止條件
    - 對於每條路徑, 當遍歷到葉子結點的時候為當前路徑的結束, 並且將當前路徑加到結果
    - if root.left == None and root.right == None:
      res.append(path)
  - 3.找出重複的子問題
    - 前序遍歷的順序是：根節點、左子樹、右子樹。
    - 對於左子樹和右子樹來說，也都是同樣的操作。
    - self.getPaths(root.left, path, res)
      self.getPaths(root.right, path, res)

時間複雜度

- 每個節點都會被訪問到，所以時間複雜度為 O(n)。
- 此外在遞迴過程中呼叫了額外的棧空間，棧的大小取決於二元樹的高度，二元樹最壞情況下的高度為 n，所以空間複雜度為 O(n)。