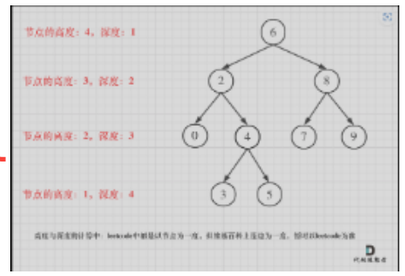
110. Balanced Binary Tree

- 平衡二元樹
  - 在二元樹中任何一個節點左右子樹的高度差不超過1
  - 判斷每一個節點左右子樹的高度 小於等於一 就符合平衡二叉樹
  - 

  - 分支主題 3
- 遞歸法
  - 前序遍歷
    - 前序的順序是中左右
    - 一直向下去遍歷 不向上返回結果
    - 求得是深度
  - 後序遍歷
    - 通過左右孩子的情況, 返回給父節點
    - 父節點再根據左右孩子的高度做加一, 層層向上返回
    - 求得是高度
  - 後序遍歷代碼實現
    - 偽代碼
      ```
      if abs(高度(root.left) - 高度(root.right)) <= 1 and root.left 也是平衡二元樹 and root.right 也是平衡二元樹:
          print('是平衡二元樹')
      else:
          print('不是平衡二元樹')
      ```
    - 遞歸法三步驟
      - 建立遞歸函數 get_hight
      - 1. 明確遞歸函數的參數和返回值
        - 參數:當前傳入的結點
        - 返回值: 以當前傳入結點為根節點的樹的高度
          - 如果當前傳入節點為根節點的二元樹已經不是二叉平衡樹了，還返回高度的話就沒有意義了。
          - 所以如果已經不是二叉平衡樹了，可以返回-1 來標記已經不符合平衡樹的規則了。
      - 2.明確的終止條件
        - 遞歸的過程中依然是遇到了空節點為終止, 返回0 表示當前節點為根節點的樹高度為0
          ```
          if not root:
              return 0
          ```
      - 3.明確單層遞歸的邏輯
        - 如何判斷以當前傳入節點為根節點的二元樹是否是平衡二元樹呢？
          當然是其左子樹高度和其右子樹高度的差值。
        - 分別求出其左右子樹的高度，然後如果差值小於等於1，則返回當前二元樹的高度，否則則返回-1，表示已經不是二叉平衡樹了。
- 迭代法
- 總結
  - 了解求二元樹深度和二元樹高度的差異
  - 求深度適合用前序遍歷
  - 求高度適合用後序遍歷