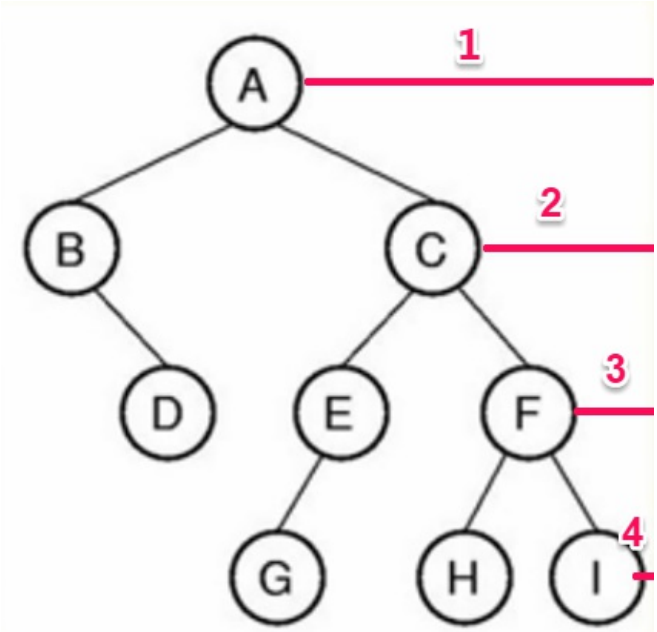
二叉树-3

depth.py

- **案例：二叉树层数**
  - **概述**：如下图所示

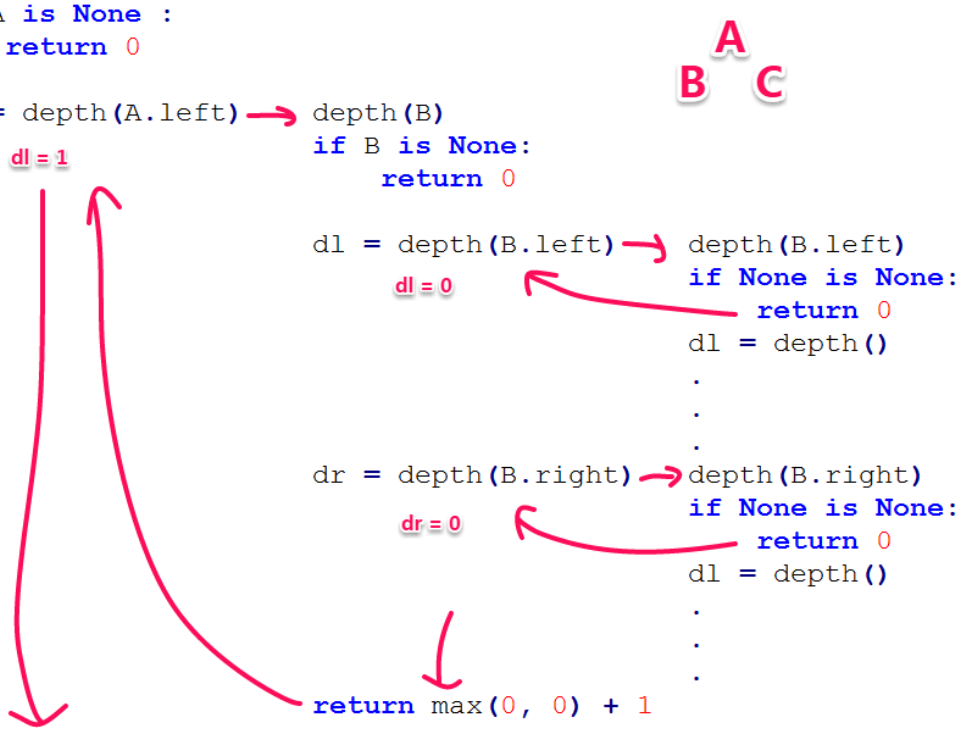


  - **思路**
    - 递归实现：二叉树的层数等于 max(左子树深度，右子树深度) + 1 。

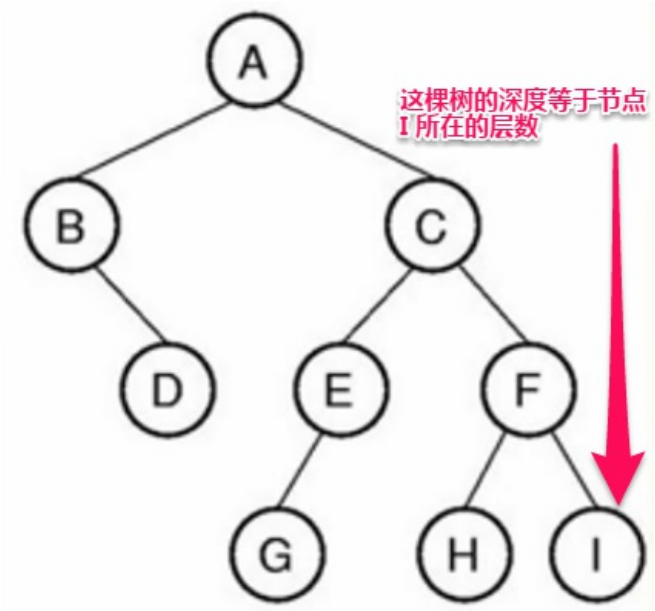
```
def depth (A):
    # 递归出口 , 所有递归都需要出口 , 而且必须写在上面
    if A is None :
        return 0

    dl = depth(A.left)
    dr = depth(node.right)

    return max(dl,  dr) + 1
```



    - 非递归实现：一棵树的层次数等于最后一个节点的所在的层数。借助层序遍历二叉树代码实现，通过最后一个出队列的节点的层数得到树的深度。



- **代码-1**：递归实现

```
def depth (node):
    # 递归出口 ，所有递归都需要出口 ，而且必须写在上面
    if node is None :
        return 0

    dl = depth(node.left)
    dr = depth(node.right)

    return max(dl, dr) + 1
```

- **代码-2**：非递归实现，Python 变量作用域

```
def depth2 (root):
    q = deque ([(root, 1)]) # 传入元素为元组 （节点名称，节点深度）

    while True:
        node , d = q.popleft()

        if node. left:
            q.append((node.left , d + 1))
        if node. right:
            q.append((node.right , d + 1))

        if not q: # 最后一个节点出队后， 队列变为空 ， 此时的 d 为最后一个节点的 d
            break

    return d # 这里为什么能够使用变量 d 呢？
```

- **And So On**