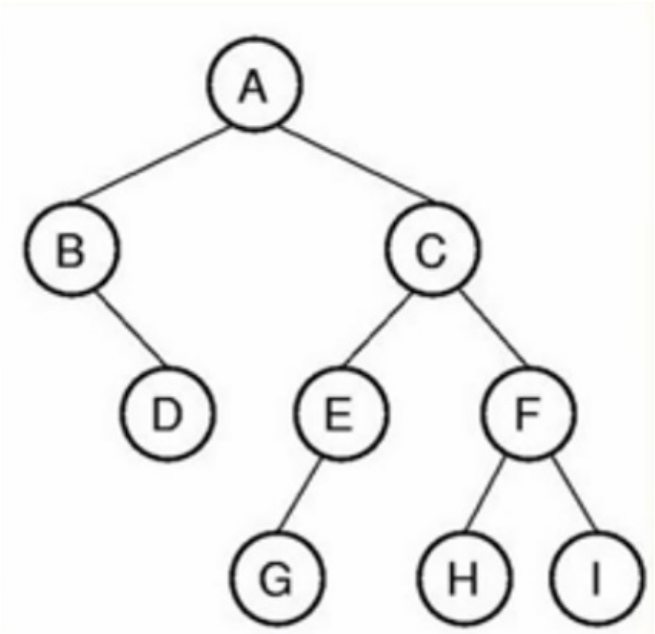
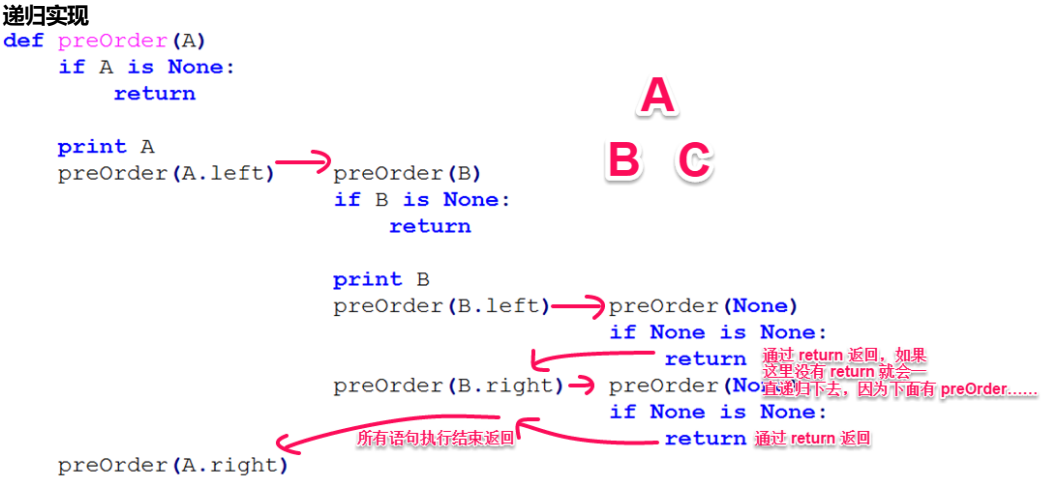
二叉树-2

iterator.py

- **案例：遍历二叉树**
  - **概述**：先、中、后都是针对根节点而言的。先序遍历是先遍历根节点、左子树、右子树。中序遍历是先遍历左子树、根节点、右子树。后续遍历是先遍历左子树、右子树、根节点。注意遍历是递归的，遍历左、右子树的时候，同样也是以先、中、后的顺序。



  - **思路**
    - **递归实现**



      - **先序遍历**：A、左子树（B、D）、右子树（C、E、G、F、H、I）。
      - **中序遍历**：左子树（B、D）、A、右子树（G、E、C、H、F、I）。
      - **后序遍历**：左子树（D、B）、右子树（G、E、H、I、F、C）、A。
    - **队列实现**：借助队列实现二叉树的层序遍历 A、B、C、D、E、F、G、H、I。根节点入队，循环这个过程（出队且将左节点、右节点入队），当队列为空时结束循环。
    - **回溯法实现**： 回溯法实现遍历二叉树
  - **代码-1**：先序遍历

```
def preOrder(node):
    # 递归出口，所有递归都需要一个出口，而且必须写在上面
    if node is None:
        return

    # 先根节点
    print node
    # 再左子树，左子树递归完才会返回
    preOrder(node.left)
```

```
    # 最后右子树
    preOrder(node.right)

if __name__ == "__main__":
    # 不要使用 print 使用 print 会最后打印出 None
    preOrder(createTree())
```

- **代码-2**：中序遍历

```
def inOrder(node):
    if node is None:
        return

    # 先左子树
    inOrder(node.left)
    # 根节点
    print node
    # 最后右子树
    inOrder(node.right)

if __name__ == "__main__":

    inOrder(createTree())
```

- **代码-3**：后序遍历

```
def postOrder(node):
    if node is None:
        return
    # 先左子树
    postOrder(node.left)
    # 再右子树
    postOrder(node.right)
    # 最后根节点
    print node

if __name__ == "__main__":

    postOrder(createTree())
```

- **代码-4**：层序遍历

```
# 自己实现，没有问题，可以有改进的地方
def levelOrder(root):
    node = root
    q = deque([node])

    while True:
        node = q.popleft()
        print node

        if not node.left == None: # if node.left: 如果 if None: 就不会进入 if 语句块儿
            q.append(node.left)

        if not node.right == None: # if node.right: 如果 if None: 就不会进入 if 语句块儿
            q.append(node.right)

        if not q:
            break
```

- **And So On**