**二叉树的深度优先遍历**

1. 实验目的：

用python书写二叉树的深度优先遍历算法

1. 实验代码：

**class** Node():  
 *# 构建一个节点类* **def** \_\_init\_\_(self, data=-1):  
 self.data = data  
 self.left = **None** self.right = **None  
  
  
class** Tree():  
 *# 构建一个树类* **def** \_\_init\_\_(self):  
 self.root = Node()  
  
 **def** add(self, data):  
 *# 给树加入节点* node = Node(data)  
 **if** self.root.data == -1: *# 如果树为空，就对根节点赋值* self.root = node  
 **else**:  
 myQueue = []  
 treeNode = self.root  
 myQueue.append(treeNode)  
 **while** myQueue: *# 对已有的节点进行遍历* treeNode = myQueue.pop(0)  
 **if not** treeNode.left:  
 treeNode.left = node  
 **return  
 elif not** treeNode.right:  
 treeNode.right = node  
 **return  
 else**:  
 myQueue.append(treeNode.left)  
 myQueue.append(treeNode.right)  
  
 **def** DFS(self, root): *# 递归实现深度优先遍历* **if** root == **None**:  
 **return** print(root.data)  
 self.DFS(root.left)  
 self.DFS(root.right)  
  
 **def** DFS\_STACK(self, root): *# 基于栈数据结构实现的深度遍历* **if** root == **None**:  
 **return** stack = []  
 stack.append(root)  
 **while** stack:  
 now\_node = stack.pop()  
 print(now\_node.data)  
 **if** now\_node.right != **None**:  
 stack.append(now\_node.right)  
 **if** now\_node.left != **None**:  
 stack.append(now\_node.left)  
   
**if** \_\_name\_\_ == **'\_\_main\_\_'**:  
 *# 主函数* datas = [1, 2, 3, 4, 5, 6, 7, 8, 9]  
 tree = Tree() *# 新建一个树对象* **for** data **in** datas:  
 tree.add(data) *# 逐个加入树的节点* print(**'递归实现深度优先遍历：'**)  
 tree.DFS\_STACK(tree.root)

1. 运行结果：

