深度优先遍历二叉树

8

10

3

19

12

6

17

9

11

利用深度优先遍历此二叉树

源代码

**import** numpy **as** np  
**import** requests  
**import** \_random  
  
*#定义二叉树中的节点***class** TreeNode(object):  
 **def** \_init\_(self,val,left=None,right=None):  
 self.val = val  
 self.left=left  
 self.right=right  
*#定义二叉树类***class** BinaryTree(object):  
 **def** \_init\_(self,root=None):  
 self.root=root  
*#先序遍历***def** preOrder(retlist,node):  
 **if** node!=None:  
 retlist.append(node)*#访问根节点* preOrder(retlist,node.left )*#遍历左子树* preOrder(retlist, node.right) *# 遍历右子树* **return** retlist  
*#中序遍历***def** inOrder(retlist,node):  
 **if** node!=None:  
 preOrder(retlist,node.left )*#遍历左子树* retlist.append(node) *# 访问根节点* preOrder(retlist, node.right) *# 遍历右子树* **return** retlist  
*#后序遍历***def** postOrder(retlist,node):  
 **if** node!=None:  
 preOrder(retlist,node.left )*#遍历左子树* preOrder(retlist, node.right) *# 遍历右子树* retlist.append(node) *# 访问根节点* **return** retlist  
  
**"""if \_\_name\_\_=='\_\_main\_\_': """  
def** main():  
 **print** ( **'----先序遍历----'**)  
 rootNode=TreeNode(11)  
 rootNode.left = TreeNode (9,left = TreeNode (6,left = TreeNode (3),right=TreeNode(8)),right=TreeNode(10))  
 rootNode.right =rootNode(17,left = TreeNode (12),right = TreeNode (19))  
 bTree=BinaryTree(rootNode)  
 ret=preOrder([],bTree .root)  
 **for** i **in** ret:  
 **print** (i.val)  
  
 **print** ( **'----中序遍历----'**)  
 ret = inOrder([],bTree .root)  
 **for** i **in** ret:  
 **print** (i.val)  
  
 **print** ( **'----后序遍历----'**)  
 ret = postOrder([],bTree .root)  
 **for** i **in** ret:  
 **print** (i.val)

实验总结：深度优先空间复杂度为线性函数，极大的降低了空间复杂度，可执行性高

实验疑惑：代码无法显示结果，仍在探索中