作业5

二叉结点、二叉树类定义：

class BinTreeNode: #二叉链表结点

def \_\_init\_\_(self,dat):

self.data=dat

self.left=None

self.right=None

def GetData(self):

return self.data

def SetData(self,item):

self.data=item

def SetLeft(self, L):

self.left=L

def SetRight(self,R):

self.right=R

class BinTrLList: #二叉链表

def \_\_init\_\_(self,A=[]): #用列表定义二叉树

L=len(A)

if L==0:

self.\_root=None

else:

if L>0:

B=[0 for i in range(L)]

for i in range(L):

if A[i]!=0:

B[i]=BinTreeNode(A[i])

for i in range(L):

if 2\*i+1<L and A[i]!=0 and A[2\*i+1]!=0:

B[i].left=B[2\*i+1]

if 2\*i+1<L and A[i]!=0 and A[2\*i+2]!=0:

B[i].right=B[2\*i+2]

self.\_root=B[0]

def IsEmpty(self):

if self.\_root==None:

return True

else:

return False

def root(self):

return self.\_root

def lchild(self):

return self.\_root.left

def rchild(self):

return self.\_root.right

def set\_root(self,rootnode):

self.\_root=rootnode

def set\_left(self,leftchild):

self.\_root.leftchild

def set\_right(self,rightchild):

self.\_root(self,rightnode)

第一题：

代码：

def IsBalaced(BinTree): #T1

def Depth(t): #计算树的高度

if t==None: return 0

else: return 1+max(Depth(t.left),Depth(t.right))

if BinTree.IsEmpty():

return 0

rheight=Depth(BinTree.\_root.right)

lheight=Depth(BinTree.\_root.left)

return abs(rheight-lheight)<=1 #判断是否平衡

第二题：

def POrder(T): #T2

if T.IsEmpty(): return

p=T.\_root

q=[p]

l=[]

cs,ps=0,1 #通过统计子节点和父节点，得知是否已经遍历了一层

i=0 #设根结点为第0层

while True:

if i%2==0: #偶数层，从左到右

p=q.pop() #从末尾出队，实现从左到右

if p.left is not None:

q.insert(0,p.left) #从头入队

cs+=1

if p.right is not None:

q.insert(0,p.right) #从头入队

cs+=1

if i%2!=0: #奇数层，从右到左

p=q.pop(0) #从头出队，实现从右到左遍历

if p.right is not None:

q.append(p.right) #从末尾进队

cs+=1

if p.left is not None:

q.append(p.left) #从末尾进队

cs+=1

l.append(p.data)

ps-=1 #实现层数的求得

if ps==0:

i+=1

ps=cs

cs=0

if q: #若队空，则退出

continue

else: break

return l

A=[7,6,15,3,0,10,17,1,4,0,0,0,12,0,20]

t=BinTrLList(A)

print POrder(t)

结果：



第三题：

代码如下：

def roadco(T,N): ##T3

if T.IsEmpty(): return

t=T.\_root

def roadcost(t,N): #递归求路径代价

if t.data==N:

print t.data

else:

if t.left !=None:

roadcost(t.left,N-t.data) #后序遍历

if t.right!=None:

roadcost(t.right,N-t.data)

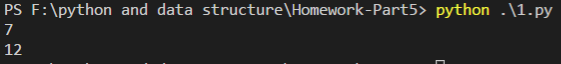
roadcost(t,N)

A=[10,5,12,7,0,0,4]

t=BinTrLList(A)

roadco(t,22) #T3

结果如下：



第四题：

def treediameter(T): ##T4

if T.IsEmpty():return 0

p=T.\_root

def Depth(t):

if t==None: return 0

else: return 1+max(Depth(t.left),Depth(t.right))

def diameter(t): #递归求直径，

if t==None: return 0

else:

ld=diameter(t.left) #左子树直径

rd=diameter(t.right) #右子树直径

lh=Depth(t.left)

rh=Depth(t.right)

return max(lh+rh+1,max(ld,rd)) #比较是否经过根节点，是则为左右子树高度和+1，否则是左右子树中的最大直径

return diameter(p)

A=[40,30,65,25,35,50,0,10,26,33,0,0,0,0,0,0,0,0,27,5,0,0,0,0,0,0,0,0,0,0,0]

t=BinTrLList(A)

print treediameter(t) #T4

结果如下：



第五题

代码如下：

def weightedroadlength(T): #T5

if T.IsEmpty(): return

p=T.\_root

q=[p]

l=[]

ps,cs,i=1,0,0

sum=0

while True: #利用层次遍历，并获得层数，和即为所在层与结点值之积的和

p=q.pop(0)

if p.left is not None:

q.append(p.left)

cs+=1

if p.right is not None:

q.append(p.right)

cs+=1

sum+=p.data\*i

ps-=1

if ps==0:

i+=1

ps=cs

cs=0

if q:

continue

else:

break

return sum

A=[10,5,12,7,0,0,4]

t=BinTrLList(A)

print weightedroadlength(t) #T5

结果如下：

第六题：

maxnode=BinTreeNode(0)

maxsum=None

def maxsubtree(T):#T6

if T.IsEmpty():return

def subtree(t): #利用递归求最大指数和

global maxsum #用全局值在递归时保存最大值和最大子树根节点

global maxnode

if t==None:

return 0

leftsum=subtree(t.left)

rightsum=subtree(t.right)

sum=t.data+leftsum+rightsum #和即为根值与左右子树之和

if maxsum<sum:

maxsum=sum

maxnode=t

return sum

t=T.\_root

subtree(t)

print "the max sum is", maxsum

print "the max root is",maxnode.data

return

A=[-4,-2,3,11,0,2,-5,-3,1,0,0,0,0,0,4]

t=BinTrLList(A)

maxsubtree(t) #T6

结果



第七题：

def keywordfind(T,b): #T7

if T.IsEmpty(): return False

a=b

t=T.\_root

tmp=a.pop(0)

while t!=None:

if tmp==t.data:

if a: tmp=a.pop(0) #若序列非空

else: break

if t.left!=None and tmp==t.left.data: #查看是否是左节点值

t=t.left

elif t.right!=None and tmp==t.right.data: #查看是否是右节点值

t=t.right

else:

return False #如果有不是则为不是关键词序列

return True

A=[40,30,65,25,35,50,0,10,26,33,0,0,0,0,0,0,0,0,27,0,0,0,0,0,0,0,0,0,0,0,0]

t=BinTrLList(A)

b=[40,30,25,26,10]

print keywordfind(t,b)

结果：

