1.

def create\_btree(tree, data):

for i in range(len(data)):

level = 0

if i == 0:

tree[level] == data[i]

else:

while tree[level]:

if data[i] > tree[level]:

level = level \* 2 + 2

else:

level = level \* 2 + 1

tree[level] = data[i]

print(i, tree)

btree = [0] \* 16

data = [10, 5, 21, 9, 13, 28, 3, 4, 1, 17, 32]

create\_btree(btree, data)

for i in range(len(btree)):

print("二元樹陣列btree[%d] = %d" % (i, btree[i]))

2.

class Node():

def \_\_init\_\_(self, data=None):

self.data = data

self.left = None

self.right = None

def insert(self, data):

if self.data:

if data < self.data:

if self.left:

self.left.insert(data)

else:

self.left = Node(data)

else:

if self.right:

self.right.insert(data)

else:

self.right = Node(data)

else:

self.data = data

def inorder(ptr):

if ptr != None:

inorder(ptr.left)

print('[%2d]' %ptr.data, end=' ')

inorder(ptr.right)

tree = Node()

datas = [10, 5, 21, 9, 13, 28, 3, 4, 1, 17, 32]

for d in datas:

tree.insert(d)

inorder(tree)

3.

class Node():

def \_\_init\_\_(self, data=None):

self.data = data

self.left = None

self.right = None

def insert(self, data):

if self.data:

if data < self.data:

if self.left:

self.left.insert(data)

else:

self.left = Node(data)

else:

if self.right:

self.right.insert(data)

else:

self.right = Node(data)

else:

self.data = data

def inorder(self):

if self.left:

self.left.inorder()

print(self.data)

if self.right:

self.right.inorder()

def depth(Node):

if Node is None:

return 0

dl = depth(Node.left)

dr = depth(Node.right)

return max(dl, dr) + 1

tree = Node()

datas = [10, 5, 21, 9, 13, 28, 3, 4, 1, 17, 32]

count = 0

for d in datas:

tree.insert(d)

count += 1

print("二元樹的節點數量為%d" %count)

print("二元樹的深度為%d" %depth(tree))