


Lab 9 : Binary Search Tree & recursion ทบทวน

- สร้าง class node ของ BST ประกอบด้วยข้อมูล 3 ส่วน คือ data left subtree และ right subtree โดยให้ค่า default ของ left subtree และ right subtree เป็น None เช่น รูป tree ข้างล่างเกิดจาก เรียก

t = node(14) 

- เขียน iterative function addi(r, d) เพื่อ add node ที่มี data d เข้าไปใน binary search tree r โดย return r กลับดังแสดงใน code ข้างล่าง

```
l = [14,4,9,7,15,3,18,16,20,5,16]
print('input: ',l)

r = None
for ele in l:
    r = addi(r, ele)

print('inorder:', end = ' ')
inOrder(r)
print()

print('printSideWay:')
printSideWay(r, 0)

print('height of ', r.data, '=', height(r));

d = 5
print('path:', d, '=', end = ' ')
path(r, d)

d = 9
t = search(r, d)
print(t.data)

d = 18
print('depth of node data ', d, '=', depth(r, d))
```

```
input: [14, 4, 9, 7, 15, 3, 18, 16, 20, 5, 16]
inorder: 3 4 5 7 9 14 15 16 16 18 20
printSideWay:
      20
     18
    16
   16
  15
 14
   9
   7
   5
  4
  3
height of 14 = 4
path: 5 = 14 4 9 7 5
9
father of 14 is None 14 is ROOT
depth of node data 18 = 2
smallest data = 3
Press any key to continue . . . _
```

- เขียน recursive function inorder(r) เพื่อพิมพ์ inorder traverse ของ tree r เพื่อทดสอบความถูกต้องของ tree ที่สร้าง
- เขียน recursive function add(r, d) เพื่อ add node ที่มี data d เข้าไปใน binary search tree r โดย return r กลับ โดยเปลี่ยน code ข้างบนให้เรียก add() แทน addi() ทดสอบโดย inorder ว่า add() ถูกต้องหรือไม่
- เขียน recursive function printSideWay(r, level) เพื่อพิมพ์ tree r หันข้าง โดยหมุนทางซ้าย 90 องศา และทดสอบความถูกต้อง โดยเทียบกับรูป tree ที่วาดด้วยมือตามลำดับการ add node ต่างๆ เข้าไป (ดังนั้นเราสามารถเรียก printSideWay(r.left, 0) กับตัวอย่างก็จะได้ left subtree ของ tree r)
- เขียน recursive function height(r) return height ของ tree r
- เขียน recursive function path(r, d) เพื่อพิมพ์ path จาก root r ไปยัง node ที่มี data d
- เขียน recursive function search(r, d) return pointer to node ที่มี data d ใน tree r
- เขียน recursive function depth(r, d) return depth ของ node ที่มี data d ใน tree r

```

class node:
    def __init__(self, data, left = None, right = None):
        self.data = data
        self.left = left if left is not None else None
        self.right = right if right is not None else None
#-----No class BST
def inOrder(r):
    if r:
        inOrder(r.left)
        print(r.data, end = ' ')
        inOrder(r.right)

def addi(r, data):
    if not r:
        return node(data)
    else:
        if data < r.data:
            r.left = addi(r.left, data)
        else:
            r.right = addi(r.right, data)
    return r

def add(r, data): # recursive add
    if not r:
        return node(data)
    else:
        if data < r.data:
            r.left = add(r.left, data)
        else:
            r.right = add(r.right, data)
    return r

def printSideWay(r, level):
    if r:
        printSideWay(r.right, level+1)
        print(' ' * 3 * level, r.data)
        printSideWay(r.left, level+1)

def height(r):
    """ return height of node pointed by r"""
    if not r:
        return -1
    else:
        hl = height(r.left)
        hr = height(r.right)
        if hl>hr:
            return hl+1
        else:
            return hr+1

def path(r, d):
    """print path from node pointed by r to node that has data d"""
    if r.data != d:
        print(r.data, end = ' ')
        if d < r.data:
            path(r.left, d)
        else:
            path(r.right, d)
    else:
        print(d)

def search(r, d):
    """return pointer to node that has data d """
    if not r: # empty tree
        return None
    if r.data == d:

```

```
        return r
    else:
        if d < r.data:
            return search(r.left, d)
        else:
            return search(r.right, d)
```