자료구조 8장 과제

Global Business & Technology
201904385 우인경

1. 순회

- 코드화면

```
1
2  from CircularQueue import *
3
4  class TNode:
5      def __init__(self,data, left, right):
6          self.data = data
7          self.left = left
8          self.right = right
9
10 def preorder(n):
11     if n is not None:
12         print(n.data, end=' ')
13         preorder(n.left)
14         preorder(n.right)
15
16 def inorder(n):
17     if n is not None:
18         inorder(n.left)
19         print(n.data, end= ' ')
20         inorder(n.right)
21
22 def postorder(n):
23     if n is not None:
24         postorder(n.left)
25         postorder(n.right)
26         print(n.data, end=' ')
27
28 def levelorder(root):
29     queue = CircularQueue()
30     queue.enqueue(root)
31     while not queue.isEmpty():
32         n = queue.dequeue()
33         if n is not None:
34             print(n.data, end = ' ')
35             queue.enqueue(n.left)
36             queue.enqueue(n.right)
37
38 def count_node(n):
39     if n is None:
40         return 0
41     else:
42         return 1 + count_node(n.left) + count_node(n.right)
43
44 def count_leaf(n):
45     if n is None:
46         return 0
47     elif n.left is None and n.right is None:
48         return 1
49     else:
50         return count_leaf(n.left) + count_leaf(n.right)
51
```

```python
def calc_height(n):
    if n is None:
        return 0
    hLeft = calc_height(n.left)
    hRight = calc_height(n.right)
    if (hLeft > hRight):
        return hLeft + 1
    else:
        return hRight + 1

d = TNode('D', None, None)
e = TNode('E', None, None)
b = TNode('B', d, e)
f = TNode('F', None, None)
c = TNode('C', f, None)
root = TNode('A', b, c)

print('\n In-Order : ', end='')
inorder(root)
print('\n Pre-Order : ', end='')
preorder(root)
print('\n Post-Order : ', end='')
postorder(root)
print('\n Level-Order : ', end='')
levelorder(root)
print()

print("노드의 개수 = %d개" % count_node(root))
print("단말의 개수 = %d개" % count_leaf(root))
print("트리의 높이 = %d" % calc_height(root))

```

- 실행화면

```
========== RESTART: /Users/inkyung/Documents/order.py

 In-Order : D B E A F C
 Pre-Order : A B D E C F
 Post-Order : D E B F C A
 Level-Order : A B C D E F
노드의 개수 = 6개
단말의 개수 = 3개
트리의 높이 = 3
>>>
```

2. 모르스 부호 구현 및 실행

- 코드화면

morse_code.py - /Users/inkyung/Documents/morse_code.py (3.9.2)

```
table = [ ('A', '.-'), ('B', '-...'), ('C', '-.-.'), ('D', '-..'), ('E', '.'), ('F', '..-.'),
          ('G', '--.'), ('H', '....'), ('I', '..'), ('J', '.---'), ('K', '-.-'), ('L', '.-..'),
          ('M', '--'), ('N', '-.'), ('O', '---'), ('P', '.--.'), ('Q', '--.-'), ('R', '.-.'),
          ('S', '...'), ('T', '-'), ('U', '..-'), ('V', '...-'), ('W', '.--'), ('X', '-..-'),
          ('Y', '-.--'), ('Z', '--..')]

class TNode:
    def __init__(self,data, left, right):
        self.data = data
        self.left = left
        self.right = right

def make_morse_tree():
    root = TNode(None, None, None)
    for tp in table:
        code = tp[1]
        node = root
        for c in code:
            if c == '.':
                if node.left == None:
                    node.left = TNode(None, None, None)
                node = node.left
            elif c == '-':
                if node.right == None:
                    node.right = TNode(None, None, None)
                node = node.right

        node.data = tp[0]
    return root

def decode(root, code):
    node = root
    for ch in code:
        if ch == '.':
            node = node.left
        elif ch == '-':
            node = node.right
    return node.data

def encode(ch):
    idx = ord(ch) - ord('A')
    return table[idx][1]


morseCodeTree = make_morse_tree()
str = input("입력 문장 : ")
mlist = []
for ch in str:
    code = encode(ch)
    mlist.append(code)
print("Morse Code : ", mlist)
print("Decoding : ", end='')
for code in mlist:
    ch = decode(morseCodeTree, code)
    print(ch, end='')
print()
```

- 실행화면

```
========== RESTART: /Users/inkyung/Documents/morse_code.py ========
입력 문장 : GAMEOVER
Morse Code :  ['--.', '.-', '--', '.', '---', '...-', '.', '.-.']
Decoding : GAMEOVER
>>>
```

3. 힙 구현 및 실행

1) 최대힙 구현

- 코드화면

```
1
2 class MaxHeap:
3     def __init__(self):
4         self.heap = []
5         self.heap.append(0)
6
7     def size(self): return len(self.heap) - 1
8     def isEmpty(self): return self.size() == 0
9     def Parent(self, i): return self.heap[i//2]
10    def Left(self, i): return self.heap[i*2]
11    def Right(self, i): return self.heap[i*2+1]
12    def display(self, msg = '힙 트리: '):
13        print(msg, self.heap[1:])
14
15    def insert(self, n):
16        self.heap.append(n)
17        i = self.size()
18        while (i != 1 and n > self.Parent(i)):
19            self.heap[i] = self.Parent(i)
20            i = i // 2
21        self.heap[i] = n
22    def delete(self):
23        parent = 1
24        child = 2
25        if not self.isEmpty():
26            hroot = self.heap[1]
27            last = self.heap[self.size()]
28            while (child <= self.size()):
29                if child < self.size() and self.Left(parent) < self.Right(parent):
30                    child += 1
31                if last >= self.heap[child]:
32                    break;
33                self.heap[parent] = self.heap[child]
34                parent = child
35                child *= 2;
36            self.heap[parent] = last
37            self.heap.pop(-1)
38        return hroot
39
40
41 heap = MaxHeap()
42 data = [2, 5, 4, 8, 9, 3, 7, 3]
43 print("[삽입 연산] : " + str(data))
44 for elem in data:
45     heap.insert(elem)
46 heap.display('[삽입 후] : ')
47 heap.delete()
48 heap.display('[삭제 후] : ')
49 heap.delete()
50 heap.display('[삭제 후] : ')
51
```

- 실행화면1

```
========== RESTART: /Users/inkyung/Documents/maxheap.py ====
[삽입 연산] : [2, 5, 4, 8, 9, 3, 7, 3]
[삽입 후] :  [9, 8, 7, 3, 5, 3, 4, 2]
[삭제 후] :  [8, 5, 7, 3, 2, 3, 4]
[삭제 후] :  [7, 5, 4, 3, 2, 3]
>>>
```

- 실행화면2

```
>>>
================ RESTART: /Users/inkyung/Documents/maxheap.py ================
[삽입 연산] : [90, 70, 80, 60, 65, 45, 40, 55, 54, 53, 52, 39, 38, 37, 36, 50]
[삽입 후] :  [90, 70, 80, 60, 65, 45, 40, 55, 54, 53, 52, 39, 38, 37, 36, 50]
[삭제한 값: 90], [삭제 후] :  [80, 70, 50, 60, 65, 45, 40, 55, 54, 53, 52, 39, 38, 37, 36]
[삭제한 값: 80], [삭제 후] :  [70, 65, 50, 60, 53, 45, 40, 55, 54, 36, 52, 39, 38, 37]
[삭제한 값: 70], [삭제 후] :  [65, 60, 50, 55, 53, 45, 40, 37, 54, 36, 52, 39, 38]
[삭제한 값: 65], [삭제 후] :  [60, 55, 50, 54, 53, 45, 40, 37, 38, 36, 52, 39]
[삭제한 값: 60], [삭제 후] :  [55, 54, 50, 39, 53, 45, 40, 37, 38, 36, 52]
[삭제한 값: 55], [삭제 후] :  [54, 53, 50, 39, 52, 45, 40, 37, 38, 36]
[삭제한 값: 54], [삭제 후] :  [53, 52, 50, 39, 36, 45, 40, 37, 38]
[삭제한 값: 53], [삭제 후] :  [52, 39, 50, 38, 36, 45, 40, 37]
>>>
```

2) 최소힙 구현

- 코드화면

```python
class MinHeap:
    def __init__(self):
        self.heap = []
        self.heap.append(0)

    def size(self): return len(self.heap) - 1
    def isEmpty(self): return self.size() == 0
    def Parent(self, i): return self.heap[i//2]
    def Left(self, i): return self.heap[i*2]
    def Right(self, i): return self.heap[i*2+1]
    def display(self, msg = '힙 트리: '):
        print(msg, self.heap[1:])

    def insert(self, n):
        self.heap.append(n)
        i = self.size()
        while (i != 1 and n < self.Parent(i)):
            self.heap[i] = self.Parent(i)
            i = i // 2
        self.heap[i] = n
    def delete(self):
        parent = 1
        child = 2
        if not self.isEmpty():
            hroot = self.heap[1]
            last = self.heap[self.size()]
            while (child <= self.size()):
                if child < self.size() and self.Left(parent) > self.Right(parent):
                    child += 1
                if last <= self.heap[child]:
                    break;
                self.heap[parent] = self.heap[child]
                parent = child
                child *= 2;
            self.heap[parent] = last
            self.heap.pop(-1)
        return hroot


heap = MinHeap()
data = [2,5,4,8,9,3,7,3]
print("[삽입 연산] : " + str(data))
for elem in data:
    heap.insert(elem)
heap.display('[삽입 후] : ')
heap.delete()
heap.display('[삭제 후] : ')
heap.delete()
heap.display('[삭제 후] : ')
heap.delete()
heap.display('[삭제 후] : ')
heap.delete()
heap.display('[삭제 후] : ')
heap.delete()
heap.display('[삭제 후] : ')
heap.delete()
```

- 실행화면

```
================= RESTART: /Users/inkyung/Documents/minheap.py ====
[삽입 연산] : [2, 5, 4, 8, 9, 3, 7, 3]
[삽입 후] :  [2, 3, 3, 5, 9, 4, 7, 8]
[삭제 후] :  [3, 5, 3, 8, 9, 4, 7]
[삭제 후] :  [3, 5, 4, 8, 9, 7]
[삭제 후] :  [4, 5, 7, 8, 9]
[삭제 후] :  [5, 8, 7, 9]
[삭제 후] :  [7, 8, 9]
>>>
```

3) 허프만 코딩트리

- 코드화면

```
from minheap import *

def make_tree(freq):
    heap = MinHeap()
    for n in freq:
        heap.insert(n)

    for i in range(0,n):
        e1 = heap.delete()
        e2 = heap.delete()
        heap.insert(e1 + e2)
        print(" (%d + %d)" %(e1, e2))

label = ['E', 'T', 'N', 'I', 'S' ]
freq = [15, 12, 8, 6, 4]
make_tree(freq)
```

- 실행화면

```
================== RESTART: /Users/inkyung/Documents/hupman.py
 (4 + 6)
 (8 + 10)
 (12 + 15)
 (18 + 27)
>>>
```