

9장 프로그래밍 실습_파이썬 코드

[프로그래밍 실습 1]

```
import collections # 큐(queue) 구조를 위한 패키지

# 이진 트리에 대한 클래스 선언
class TreeNode:
    def __init__(self):
        self.data = 0
        self.left = None
        self.right = None

# 이진 트리의 높이를 계산하는 함수
def height(root, node_values, level_labels):
    ans = 0
    queue = collections.deque()
    # 이진 트리의 노드가 하나도 없을 때 0을 반환
    if root is None:
        return ans

    queue.append(root) # 루트 노드를 큐에 삽입

    # 선언된 큐에 노드가 있는 동안 계속 반복
    while queue:
        currSize = len(queue) # 현재의 큐 크기를 대입
        # 큐의 currSize개의 노드들을 디큐
        while currSize > 0:
            currNode = queue.popleft() # 탑 노드를 디큐
            currSize -= 1 # currSize를 감소시킴
            # 디큐된 노드에 ans를 레벨로 대입
            level_labels[node_values[currNode.data]] = ans
            # 디큐된 노드의 왼쪽 자식 노드와 오른쪽 자식 노드가 있으면 인큐
            if currNode.left is not None:
                queue.append(currNode.left)
            if currNode.right is not None:
                queue.append(currNode.right)

        ans += 1 # 레벨값을 증가
    return ans

# 이진 트리를 구성하는 재귀함수
def insert(x, p):
    if p == None:
        p = TreeNode()
        p.data = x
    elif x < p.data:
        p.left = insert(x, p.left)
    else:
        p.right = insert(x, p.right)
    return p

# 트리의 노드 라벨과 각 노드에 대한 값을 나타내는 사전 데이터 구조
node_values = {
    10 : "v1",
    5 : "v2",
```

[illegible]

```
# 마지막으로 이진 트리의 높이를 출력
print(end="\b\b이다. 그리고 트리의 높이는 ")
print(height_of_tree, "이다." , sep = "", end = "\n\n")
```

[프로그래밍 실습 2]

```
from __future__ import print_function
from sys import stdin

def printf(str, *args):
    print(str % args, end='')

MAX_VERTICES = 9          # 최대 정점의 수
MAX_EDGES = 16            # 최대 간선의 수
INF = 1000                # 셀프 루프를 갖지 않도록 사용될 값

# 그래프를 저장하기 위한 클래스 선언
class Dist:
    def __init__(self):
        self.dist = [0 for i in range(MAX_VERTICES)]
        self.edge = [['' for j in range(2)] for i in range(MAX_VERTICES)]

# 입력될 간선과 가중치를 저장하기 위한 간선 리스트
weight = [[0 for j in range(MAX_VERTICES)] for i in range(MAX_VERTICES)]
# 각 정점에 대한 방문 여부를 알기 위한 배열
selected = [False for j in range(MAX_VERTICES)]
dists = Dist()

# 방문하지 않는 정점들 중 최단 거리를 갖는 정점을 찾아주는 함수
def get_min_vertex(n):
    v = -1                # 방문하지 않는 정점을 위한 변수
    # 0~n의 정점들 중 방문하지 않는 첫 번째 정점 찾을
    for i in range(0, n):
        if not selected[i]:
            v = i
            break
    # 최단 거리를 갖는 정점 v를 찾을
    for i in range(0, n):
        # 만약 정점 i를 방문하지 않고 정점 i까지의 최단 거리가 정점 v까지의 최단 거리보다 작으면
        if (not selected[i]) and (dists.dist[i] < dists.dist[v]):
            v = i          # 최단 거리를 갖는 정점 v를 갱신
    return v              # 최단 거리를 갖는 정점 v를 반환

# 시작 정점 s부터 나머지 모든 정점들까지의 최단 거리를 계산하는 함수
def prim(s, n):
    printf("\t\t\t")
    # 최단 거리를 갖는 dists.dist와 정점 방문에 대한 배열을 초기화
    for u in range(0, n):
        dists.dist[u] = INF
        selected[u] = False

    # 시작 정점에 대한 최단 거리를 0으로 선언
    dists.dist[s] = 0

    for i in range(0, n):
        u=get_min_vertex(n)          # 최단 거리를 갖고 아직 방문하지 않는 정점
        selected[u] = True           # 정점 u를 방문했다고 표시
        # 정의에 따라 연결되지 않는 정점이 있을 경우 최소 스패닝 트리를 찾을 수 없음
```

[illegible]

```

        start = u_temp - 97
        u = u_temp - 97
        v = v_temp - 97
        weight[u][v] = temp          # u-v 간선의 가중치 대입
        weight[v][u] = temp          # v-u 간선의 가중치 대입

    printf("\n\t\t\t그러므로 %c로부터 최소 스패닝 트리는 다음과 같다.\n"%chr(start + 65))
    prim(start,MAX_VERTICES)         # 최소 스패닝 트리를 계산하는 prim 함수 호출
    print("\n\n")

stdin.readline()

```

[프로그래밍 실습 3]

```

from __future__ import print_function
from sys import stdin

def printf(str, *args):
    print(str % args, end='')

# 이진 트리에 대한 클래스 선언
class Node:
    def __init__(self):
        self.left = None
        self.data = 0
        self.right = None

# 전위 탐색에 대한 재귀함수
def preorder(p):
    if p != None:
        printf("%3d \u2192 "%p.data)
        preorder(p.left)
        preorder(p.right)
    return 0

# 중위 탐색에 대한 재귀함수
def inorder(p):
    if p != None:
        inorder(p.left)
        printf("%3d \u2192 "%p.data)
        inorder(p.right)
    return 0

# 후위 탐색에 대한 재귀함수
def postorder(p):
    if p != None:
        postorder(p.left)
        postorder(p.right)
        printf("%3d \u2192 "%p.data)
    return 0

# 이진 트리의 노드들을 입력하기 위한 함수
def insert(x, p):
    if p == None:
        p = Node()
        p.data = x
    elif x < p.data:

```

```

    p.left = insert(x, p.left)
else:
    p.right = insert(x, p.right)
return p

```

```
prn = None
```

해당 프로그램에 대한 메뉴 및 안내를 출력

[illegible][illegible]

메뉴의 선택번호가 5가 될 때까지 반복

```
while True:
```

```

printf("\n\t\t\t메뉴를 입력하라.: ")
values = stdin.readline().strip('\n').split() # 메뉴 선택번호 입력
if len(values) > 0:
    ch = int(values[0])
    if ch == 1: # 입력된 이진 트리에 대한 전위 탐방 출력
        print("\t\t\t전위 탐방: ", end = "")
        preorder(prn) # 전위 탐방 함수 호출
        print(end="\b\b\b\t\t\t\t\t\n")
    elif ch == 2: # 입력된 이진 트리에 대한 중위 탐방 출력
        print("\t\t\t중위 탐방: ", end = "")
        inorder(prn) # 중위 탐방 함수 호출
        print(end="\b\b\b\t\t\t\t\t\n")
    elif ch == 3: # 입력된 이진 트리에 대한 후위 탐방 출력
        print("\t\t\t후위 탐방: ", end = "")
        postorder(prn) # 후위 탐방 함수 호출
        print(end="\b\b\b\t\t\t\t\t\n")
    elif ch == 4: # 이진 트리를 입력받음
        printf("\n\t\t\t노드 값 몇 개를 입력하라.: ")
        values = stdin.readline().strip('\n').split()
        count = 0
        while len(values) > count:
            x = int(values[count])
            prn = insert(x,prn)
            count += 1
    elif ch == 5: # 프로그램 종료
        break

```

[illegible]