

8장 프로그래밍 실습_파이썬 코드

[프로그래밍 실습 1]

```
from __future__ import print_function
from sys import stdin

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

domain = 5      # 집합의 크기

# readfile 함수가 텍스트 파일에서 관계를 읽고 adjacency에 대입
def readfile(fn, adjacency):
    adjacencymx = [[0 for j in range(len(adjacency[0]))] for i in range(len(adjacency))]
    fp = open(fn, 'r')
    printf("파일로 입력받은 관계 R은 다음과 같다.\n")
    print("R = {", end = "")

    # 줄별로 파일을 읽고 lines에 대입
    lines = fp.readlines()

    # lines에서의 관계를 adjacency에 대입하고 A에 대한 함수 출력
    for line in lines:
        values = line.strip('\n').split()
        x = int(values[0])
        y = int(values[1])
        adjacencymx[x][y] = 1
        adjacencymx[y][x] = 1
        printf("(%d, %d)," % (x,y))
    print("\b}");
    fp.close()
    return adjacencymx

# 관계행렬을 위한 2차원 배열 정의 및 초기화
adjacencymx = [[0 for j in range(domain+1)] for i in range(domain+1)]
fn = "pp8-1.dat"      # 파일 이름

# 파일을 읽고 데이터를 adjacencymx에 대입
adjacencymx = readfile(fn,adjacencymx)
printf("\n\n")

# 각 정점에 대한 차수 계산 출력
printf("%5c각 정점에 대한 차수가 다음과 같다.\n\n%" ' ')

for i in range(1, domain + 1):
    degree = 0      # 차수를 위한 변수
    # 정점 i에 대한 차수 계산
    for j in range(1, domain + 1):
        if adjacencymx[i][j] == 1:
            degree += 1
    if i == domain: # 마지막 정점의 경우
        printf("%5c정점 %d의 차수는 %2d이다.\n" %(' ',i,degree))
    else:
        printf("%5c정점 %d의 차수는 %2d,\n" %(' ',i,degree))
```

```

# 관계행렬 출력
printf("\n\n%c관계 R을 인접행렬로 표현하면 다음과 같다.\n\n%" ' ')

print("          ", end = "")
for i in range(1, domain + 1):
    print(i, end = " ")
print()
print("          \u250c          \u2510")
for i in range(1, domain + 1):
    print("          ", i, "| ", end = "")
    for j in range(1, domain + 1):
        print("%d" % adjacencymx[i][j], end = " ")
    print("| ")
print("          \u2514          \u2518")

stdin.readline()

```

[프로그래밍 실습 2]

```

# 영어 소문자로 표현된 정점들을 인덱스값과 매핑하기 위한 자료구조
dict = {
    "a" : 0,
    "b" : 1,
    "c" : 2,
    "d" : 3,
    "e" : 4,
    "f" : 5,
    "g" : 6
}

# 인접행렬을 위한 2차원 배열 정의 및 초기화
adj_mat = [[0] * len(dict) for _ in range(len(dict))]

# 입력한 간선들을 포함한 2차원 배열
input = [["a","b"],["a","c"],["b","c"],["c","d"],["d","e"],["d","f"],["d","g"],["e","f"]]
key_list = list(dict.keys())      # dict의 키값들을 대입
val_list = list(dict.values())    # dict의 값들을 대입

# 그래프 구조를 위한 Graph Class 선언
class Graph:
    adj = []      # 입력 그래프를 저장하기 위한 빈 배열

    # 정점의 수가 v이고 간선의 수가 e인 빈 그래프를 선언하고 초기화하는 멤버 함수
    def __init__(self, v, e):
        self.v = v
        self.e = e
        Graph.adj = [[0 for i in range(v)]
                      for j in range(v)]

    # 간선을 추가하는 함수
    def addEdge(self, start, e):
        Graph.adj[start][e] = 1
        Graph.adj[e][start] = 1

    # 깊이 우선 탐색 함수
    def DFS(self, start, visited):

        print(key_list[start], end = "")

```

```

print( end = " -> ")
visited[start] = True

# 그래프에서 탐색되지 않은 정점에 대해 DFS() 함수를 재귀적으로 호출
for i in range(self.v):
    if (Graph.adj[start][i] == 1 and
        (not visited[i])):
        self.DFS(i, visited)

# 너비 우선 탐색 함수
def BFS(self, start):
    visited = [False] * self.v
    q = [start]      # bfs를 적용하기 위한 큐 구조를 선언하고 start 정점을 삽입

    visited[start] = True    # start 정점을 탐색했다고 표시

    while q:
        vis = q[0]          # 큐에 있는 정점과 탐색에 있는 정점을 vis에 대입

        # vis를 출력하고 큐에서 팝
        position1 = val_list.index(vis)
        print(key_list[position1], end = "")
        print( end = " -> ")
        q.pop(0)

        for i in range(self.v):
            # vis의 탐색되지 않은 인접한 정점들을 큐에 입력
            if (Graph.adj[vis][i] == 1 and
                (not visited[i])):
                q.append(i)
                visited[i] = True    # 정점 i를 탐색했다고 표시

# 새로운 그래프 G 선언
G = Graph(7, 8)

# 선언된 그래프 G를 입력 데이터를 통해 초기화
for pair in input:
    G.addEdge(dict[pair[0]],dict[pair[1]])

# 그래프 G에 대한 인접행렬을 출력
print("예제로 제공된 그래프를 인접행렬로 표현하면 다음과 같다.")

dict_keys = list(dict.keys())
print(end = "          ")
for i in dict:
    print(i, end = " ")
print()

print("          \u250c          \u2510")
for i in range(0, len(dict)):
    print("          ", dict_keys[i], "| ", end = "")
    for j in range(0, len(dict)):
        print("%d" % G.adj[i][j], end = " ")
    print("| ")
print("          \u2514          \u2518")

# 너비 우선 탐색 방법으로 탐색한 정점들을 순서대로 출력
print("\n그래프를 너비 우선 탐색 방법으로 탐색할 때 정점들의 순서는 다음과 같다.: \n")
G.BFS(dict["a"])

```

```

print(end="\b\b\b      \n")

# 정점들의 탐색에 대한 정보를 저장하기 위한 이진 배열 선언 및 초기화
visited = [False for i in range(len(dict))]

# 깊이 우선 탐색 방법으로 탐색한 정점들을 순서대로 출력
print("\n\n그래프를 깊이 우선 탐색 방법으로 탐색할 때 정점들의 순서는 다음과 같다.: \n")
G.DFS(dict["a"], visited)
print(end="\b\b\b      \n\n\n")

```

[프로그래밍 실습 3]

```

# C 언어 코드의 getch()는 파이썬에서 stdin.readline()으로 대체함
from __future__ import print_function
from sys import stdin

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

maxvalue = 5    # 정점의 수
n = 6          # 입력할 간선의 수

# 입력할 간선들을 저장하기 위한 (nx2) 이차원 배열 선언 및 초기화
b = [[0 for j in range(3)] for i in range(n+1)]
# 입력할 간선들을 저장하기 위한 (nxn) 인접행렬 선언 및 초기화
a = [[0 for j in range(maxvalue+1)] for i in range(maxvalue+1)]

order = [0 for i in range(maxvalue+1)] # 차수 기준으로 정점들을 내림차순으로 저장할 배열 선
언 및 초기화
deg = [0 for i in range(maxvalue+1)] # 각 정점의 차수에 대한 배열 선언 및 초기화
col = [0 for i in range(maxvalue+1)] # 각 정점의 색에 대한 배열 선언 및 초기화

# 5개 정점으로 구성된 그래프에 대한 간선들을 입력받고 a와 b에 대입
printf("정점이 5개인 그래프이다(정점의 라벨은 1~5이다.)\n")
printf("간선을 1 2와 같은 순서쌍으로 6개 입력하라.\n")
for i in range(1, n+1):
    values = stdin.readline().strip('\n').split()
    b[i][1] = int(values[0])
    b[i][2] = int(values[1])
    a[b[i][1]][b[i][2]] = 1
    a[b[i][2]][b[i][1]] = 1

# 각 정점에 대한 차수를 계산해서 배열 deg에 대입
for i in range(1, maxvalue+1):
    for j in range(1, maxvalue+1):
        if a[i][j] != 0:
            deg[i] = deg[i] + 1

# 계산된 차수의 값들을 출력
printf("\n\n%c각 정점에 대한 차수가 다음과 같다.\n\n%" ' ')

for i in range(1, maxvalue+1):
    if i == maxvalue:
        printf("%5c정점 %d의 차수는 %2d이다.\n" %(' ', i, deg[i]))
    else:
        printf("%5c정점 %d의 차수는 %2d,\n" %(' ', i, deg[i]))

```

```

# order 배열을 초기화
for i in range(1, maxvalue+1):
    order[i] = i

# 정점들을 차수에 따라 내림차순으로 정렬
for i in range(1, maxvalue):
    for j in range(i+1, maxvalue+1):
        if deg[order[i]] < deg[order[j]]:
            temp = order[i]
            order[i] = order[j]
            order[j] = temp

# 각 정점을 착색하는 반복문
for i in range(1,maxvalue + 1):
    v = order[i]

    # 현재 정점에서 착색되지 않은 최소의 색을 찾음
    usedColors = [False for i in range(maxvalue+1)]
    for adjVertex in range(1, maxvalue + 1):
        if a[v][adjVertex] == 1 and col[adjVertex] != 0:
            usedColors[col[adjVertex]] = True

    # 사용하지 않는 최소의 색을 현재 정점에 대입
    for c in range(1, maxvalue + 1):
        if not (usedColors[c]):
            col[v] = c
            break

# 색의 수를 세기
col_number = -1
for i in range(1, maxvalue):
    if col[i] >= col_number:
        col_number = col[i]

# 착색된 그래프에 대한 정보를 출력
printf("\n%5c정점을 착색한 결과는 다음과 같다.\n\n%" ' ')

for colors in range(1, col_number + 1):
    print("%5c정점 '%" ' ', end = "")
    for i in range(1, maxvalue+1):
        if col[i] == colors:
            print(i, end = ",")
    print(end="\b(들)이")
    print(" %d번째 색으로 착색됨" %colors)

printf("\n%5c이 그래프는 %d색 가능하다.\n\n%"(' ',col_number));

```