

5장 프로그래밍 실습_파이썬 코드

[프로그래밍 실습 1]

```
A = range(1,11) # 집합 A의 정의 및 초기화
B = list(map(chr, range(ord('a'), ord('h')+1))) # 집합 B의 정의 및 초기화

# setout 함수가 집합을 출력
def setout(A):
    if len(A) == 0:
        print("\u2205")
        return
    print("{ ", end = "")
    for i in range(0, len(A)):
        if(i < len(A) - 1):
            print(A[i], " ", end = "", sep = "")
        print(A[i], "}\n")

# 집합 A를 출력
print("A = ", end = "")
setout(A)

# 집합 B를 출력
print("B = ", end = "")
setout(B)

# A와 B의 곱집합 출력
print("\nA \u2a2f B = { ", end = "")
for i in A:
    for j in B:
        print("(%d, %c), " %(i,j), end = " ")
    print("\b\b } \n")
```

[프로그래밍 실습 2]

```
from sys import stdin
import os

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

MAXITEM = 10 # 관계행렬의 최대 크기를 위한 상수
# 관계행렬의 정의 및 0으로 초기화
m = [[0 for col in range(MAXITEM)] for row in range(MAXITEM)]

# 사용자로부터 관계행렬의 크기와 관계행렬을 입력받기
maxvalue = int(input("입력으로 넣을 관계행렬의 크기를 입력하라.: "))
printf("\n")
print("1과 0으로 구성된 ", maxvalue, "x", maxvalue, "인 관계행렬을 입력하라.: ", sep = '')

# 한 줄씩 입력될 행을 리스트에 저장
list = []
while len(list) < maxvalue*maxvalue:
    list += stdin.readline().split()
```

```

# 문자열로 저장된 리스트를 m에 정수형으로 대입
for i in range(1, maxvalue+1):
    for j in range(1, maxvalue+1):
        m[i][j] = int(list[(i-1)*maxvalue + (j-1)])

print("\n주어진 관계행렬에 대해 길이가 ", 3, "인 경로(들)의 리스트는 다음과 같다.: \n", sep =
'')
count = 0

# 행렬 m에 대해 길이가 3인 경로를 계산하면서 출력
for i in range(1, maxvalue+1):
    for j in range(1, maxvalue+1):
        if m[i][j] == 1:
            for x in range(1, maxvalue+1):
                if m[j][x] == 1:
                    for y in range(1, maxvalue+1):
                        if m[x][y] == 1:
                            printf("(%2d%2d%2d%2d) => (%d..%d)\n" %(i,j,x,y,i,y))
                            count = count + 1

# 행렬 m에 길이가 3인 경로가 없으면:
if count == 0:
    os.system('cls')
    print("\n주어진 관계행렬에 대해 길이가 ", 3, "인 경로가 없다.\n", sep = '')
print()

```

[프로그래밍 실습 3]

```

from __future__ import print_function
from sys import stdin

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

domain = 20          # 관계행렬의 크기를 위한 상수

# matrixout 함수가 행렬 mx를 출력
def matrixout(mx):
    print()
    print("\u250c                                     \u2510")

    for i in range(0, domain):
        print("| ", end = "")
        for j in range(0, domain):
            print("%2.f" % mx[i][j], end = " ")
        printf(" |\n")
        print("\u2514                                     \u2518")

# 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')
    # 집합 A를 출력
    print("A = { ", end = "")
    for i in range(1, domain + 1):
        print(i, end = ", ")
    print("\b\b }\n")

```

```

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

# lines에서의 관계를 adjacency에 대입하고 집합 A에 대한 관계 R을 출력
print("R = { ", end = "")
for line in lines:
    values = line.strip('\n').split()
    x = int(values[0])
    y = int(values[1])

    adjacencymx[x][y] = 1

    print("(%2d, %2d)," % (x,y), end = " ")
print("\b\b }")
fp.close()
return adjacencymx

fn = "pp5-3.dat"            # 텍스트 파일 이름
# 관계(부울)행렬을 위한 2차원 배열 정의 및 초기화
adjacencymx = [[0 for j in range(domain+1)] for i in range(domain+1)]

print()
# 텍스트 파일에서 읽은 관계들로 이루어진 관계행렬을 대입
adjacencymx = readfile(fn, adjacencymx)

matrixout(adjacencymx) # printing matrix

printf("\n\n")

reflexivity = 1
# 관계행렬이 반사관계 여부를 확인
for i in range(1, domain+1):
    # iRi가 없으면 바로 반복문을 멈춤
    if adjacencymx[i][i] == 0:
        reflexivity = 0
        break

# 반사관계 여부를 출력
if reflexivity == 1:
    printf("%2cR은 반사관계이다.\n%" ' ')
else:
    printf("%2cR은 반사관계가 아니다.\n%" ' ')

symmetry = 1
test = 1
# 관계행렬이 대칭관계 여부를 확인
for i in range(1, domain+1):
    for j in range(1, domain+1):
        # iRj가 있을 때 jRi가 없으면 바로 반복문을 멈춤
        if adjacencymx[i][j] == 1 and adjacencymx[j][i] == 0:
            symmetry = 0
            test = 0
            break
    if test == 0:
        break

# 대칭관계 여부를 출력
if symmetry == 1:
    printf("%2cR은 대칭관계이다.\n%" ' ')
else:

```

```

printf("%2cR은 대칭관계가 아니다.\n%" ' ')

transitivity = 1
test = 1
# 관계행렬이 추이관계 여부를 확인
for i in range(1, domain+1):
    for j in range(1, domain+1):
        if adjacencymx[i][j] == 1:
            for k in range(1,6):
                # iRj와 jRk가 있을 때, iRk가 없으면 바로 반복문을 멈춤
                if adjacencymx[j][k] == 1 and adjacencymx[i][k] == 0:
                    transitivity = 0
                    test = 0
                    break
            if test == 0:
                break

    if test == 0:
        break

# 추이관계 여부를 출력
if transitivity == 1:
    printf("%2cR은 추이관계이다.\n%" ' ')
else:
    printf("%2cR은 추이관계가 아니다.\n%" ' ')

print()

```

[프로그래밍 실습 4]

```

from sys import stdin
from sys import stdin

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

# setout 함수가 이항관계들로 이루어진 집합을 출력
def setout(row, col, size):
    r = 0
    c = 0
    print("R = { ", end = "")

    for r in range(0, size):
        print("(%2d, %2d)," %(row[r],col[c]), end = " ")
        c += 1
    print("\b\b }")

# matrixout 함수가 정방행렬 mx를 출력
def matrixout(mx, size):
    blanks = size * 2 + 2

    print("\u250c", end = "")
    for i in range(0,blanks):
        print(" ", end = "")
    print("\u2510")

    for i in range(0, size):

```

```

print("{", end = "")
for j in range(0, size):
    print("%d" % mx[i][j], end = " ")
print("}")

print("\u2514", end = "")
for i in range(0, blanks):
    print(" ", end = "")
print("\u2518")

# small_mat 함수가 크기가 50보다 작은 관계행렬을 계산 후 단계별로 출력 진행
def small_mat(domain):
    # 관계행렬을 위한 2차원 배열 정의 및 초기화
    Mrs = [[0 for j in range(0, domain)] for i in range(0, domain)]
    i = 0
    print("집합 A에 대한 관계들을 (멈추기 위해 -1 -1)을 입력하라.: ")
    # -1 -1이 입력될 때까지 사용자로부터 관계를 입력받으면서 Mrs에 대입
    while True:
        values = stdin.readline().strip('\n').split()
        i = int(values[0])
        j = int(values[1])
        if i == -1:
            break
        else:
            Mrs[i-1][j-1] = 1
    # 관계행렬 전체를 출력
    printf("\n[ 관계행렬 ]\n")
    matrixout(Mrs, domain)
    printf("\n")

    input("\t\t\t<<아무 키나 입력하라.>>")
    printf("\n")

# 와샐 알고리즘을 계산해서 각 단계가 끝나면 수정된 행렬을 출력
for i in range(0, domain):
    for j in range(i+1, i+domain):
        for k in range(i+1, i+domain):
            m = j%domain
            n = k%domain
            if Mrs[i][n] == 1 and Mrs[m][i] == 1:
                Mrs[m][n] = 1
            printf("[ W%d ] \n"%(i+1))
            matrixout(Mrs, domain) # 수정된 행렬을 출력
            printf("\n")
            # 관계를 출력
            for s in range(0, domain):
                for t in range(0, domain):
                    if Mrs[s][t] == 1:
                        printf("(%d, %d) " %(s+1, t+1))

                printf("\n")
            printf("\n\n")

# 사용자를 위해 안내문 출력
print("\n\t\t\t\u250c-----\u2510")
print("\t\t\t|          와샐 알고리즘          |")
print("\t\t\t\u2514-----\u2518\n\n")

# 관계행렬의 크기를 입력받음

```

```
size = int(input("\n집합 A가 1부터 n까지의 자연수이므로 자연수 n을 입력하라.: "))
```

```
# 크기가 0이거나 0보다 작을 때 프로그램을 멈춤
```

```
if size == 0:  
    exit()
```

```
if size <= 50:  
    small_mat(size)
```

```
else : # 집합 A의 크기가 50보다 클 때
```

```
    org = size
```

```
    print("집합 A에 대한 관계들을 (멈추기 위해) -1 -1을 입력하라.: ")
```

```
    # 큰 행렬을 저장하기 위한 배열 2개를 선언
```

```
    row = []
```

```
    col = []
```

```
    count = 0
```

```
    # -1 -1이 입력될 때까지 사용자로부터 관계를 입력받으면서 Mrs에 대입
```

```
    while True:
```

```
        pair = stdin.readline().strip('\n').split()
```

```
        if int(pair[0]) == -1 or int(pair[1]) == -1:  
            break
```

```
        row.append(int(pair[0]))
```

```
        col.append(int(pair[1]))
```

```
        count += 1
```

```
    k = 0
```

```
    set_row = [None] * org # 행렬의 특정 행을 저장하기 위한 배열
```

```
    set_col = [None] * org # 행렬의 특정 열을 저장하기 위한 배열
```

```
# 크기가 50보다 큰 관계행렬일 때 큰 행렬을 출력하지 않고 결과만 출력
```

```
    while k < org:
```

```
        temp_count = count
```

```
        index_row = 0
```

```
        index_col = 0
```

```
    # 해당 반복문으로 k번째 행과 열에서 1이 있는 인덱스를 저장
```

```
    for i in range(0, temp_count):
```

```
        if row[i] == k:
```

```
            set_col.append(col[i]) # k번째 행을 set_col에 저장
```

```
            index_col += 1 # k번째 행에 1 있는 셀을 세기
```

```
        if col[i] == k:
```

```
            set_row.append(row[i]) # k번째 열을 set_row에 저장
```

```
            index_row += 1 # k번째 열에 1 있는 셀을 세기
```

```
    for c in range(0, index_col):
```

```
        for r in range(0, index_row):
```

```
            i = 0
```

```
            # 집합 set_row x set_col의 관계가 원래 행렬에 존재하는지 확인
```

```
            for i in range(0, temp_count):
```

```
                if row[i] == set_row[r] and col[i] == set_col[c]:
```

```
                    break
```

```
            # 관계 (set_row[r], set_col[c])가 행렬에 없으면 row와 col에 추가
```

```
            if i == temp_count:
```

```
                row.append(set_row[r])
```

```
                col.append(set_col[c])
```

```
                count += 1
```

```
    k += 1
```

```
# 마지막 관계에 대한 출력
```

```
print("와살 알고리즘을 진행한 결과에 대한 관계 R은 다음과 같다.: ")  
setout(row, col, count)
```