Department of Information and Communication Engineering

Pabna University of Science and Technology

Course: Discrete Mathematics Sessional (PART-A)

Course code: ICE-2106 (PART-A)

Problem #01:

Let A be the set {1, 2, 3, 4}. Write a program to find the ordered pairs are in the relation $\mathbf{R1} = \{(\mathbf{a}, \mathbf{b}) \mid \mathbf{a\ divides\ b}\}$ $\mathbf{R2} = \{(\mathbf{a}, \mathbf{b}) \mid \mathbf{a} \leq \mathbf{b}\}$

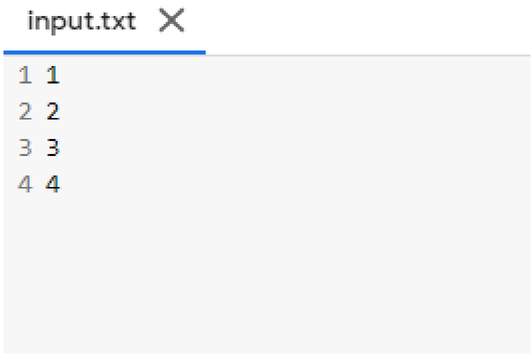
//input.txt

1

2

3

4



```
1
2
3
4
```

```
from itertools import product

with open("/content/sample_data/input.txt", "r", encoding="utf-8") as g:
 S = list(map(int, g.readlines()))
print("S= "+str(S))

res=[(i,j) for i,j in product(S,repeat=2) if i%j==0 or j%i==0]
res2=[(i,j) for i,j in product(S,repeat=2) if i<=j]
# printing result
print ("The pair list is for a/b : " + str(res))
print ("The pair list is for a<=b : " + str(res2))
```

Output:

```
S= [1, 2, 3, 4]
The pair list is for a/b : [(1, 1), (1, 2), (1, 3), (1, 4), (2, 1), (2, 2),
(2, 4), (3, 1), (3, 3), (4, 1), (4, 2), (4, 4)]
The pair list is for a<=b : [(1, 1), (1, 2), (1, 3), (1, 4), (2, 2), (2, 3),
(2, 4), (3, 3), (3, 4), (4, 4)]
```

Problem #02:

Suppose that $\mathrm{A} = \{1, 2, 3\}$ and $\mathrm{B} = \{1, 2\}$. Let R be the relation from A to B containing (a, b) if $\mathrm{a} \in \mathrm{A}$, $\mathrm{b} \in \mathrm{B}$ and $\mathrm{a} > \mathrm{b}$. Write a program to find the relation R and also represent this relation in matrix form.

```
import numpy as np
with open("/content/sample_data/input.txt", "r", encoding="utf-8") as g:
 list1 = list(map(int, g.readlines()))
with open("/content/sample_data/input.txt", "r", encoding="utf-8") as g:
 list2 = list(map(int, g.readlines()))

# using list comprehension
output = [(a, b) for a in list1
          for b in list2 if a > b]
output2 = [1 if a>b else 0 for a in list1
          for b in list2]

data = np.array(output2).reshape(4,4)

print(output)
print(data)
```

Output:

```
[(2, 1), (3, 1), (3, 2), (4, 1), (4, 2), (4, 3)]
[[0 0 0 0]
 [1 0 0 0]
 [1 1 0 0]
 [1 1 1 0]]
```

Problem #03: *Suppose that the relations R1 and R2 on a set A are represented by the matrices*

$M_{R1} = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix}$ *and* $M_{R2} = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \\ 1 & 1 & 0 \end{bmatrix}$. *Write a program to find the* $M_{R1 \cup R2}$ *and* $M_{R1 \oplus R2}$.

```
def matrix_intersection(mat1, mat2):
    rows = len(mat1)
    cols = len(mat1[0])
    print('Rows=', rows, 'Cols=', cols)
    mat_inter = []
    for i in range(len(mat1)):
        mat_inter.append([mat1[i][j] and mat2[i][j] for j in
range(len(mat1[0]))])

    return mat_inter
```

```
def matrix_union(mat1, mat2):
    mat_union = []
    for i in range(len(mat1)):
        mat_union.append([mat1[i][j] or mat2[i][j] for j in
range(len(mat1[0]))])

    return mat_union



matrix1 = [[1, 0, 1],
           [1, 0, 0],
           [0, 1, 1]]
matrix2 = [[1, 0, 1],
           [0, 1, 1],
           [1, 0, 1]]

# print('Matrix Intersection', mat_inter)
print('First Matrix=', matrix1)
print('Second Matrix=', matrix2)

mi = matrix_intersection(matrix1, matrix2)
print('Matrix Intersection', mi)

mu = matrix_union(matrix1, matrix2)
print('Matrix Union', mu)
v = ['p', 'q', 'r']

r1 = []
for i in range(len(mi)):
    for j in range(len(mi[0])):
        if mi[i][j] == 1:
            r1.append((v[i], v[j]))

print(r1)

r2 = []
for i in range(len(mu)):
    for j in range(len(mu[0])):
        if mu[i][j] == 1:
            r2.append((v[i], v[j]))

print(r2)
```

Output:

```
First Matrix= [[1, 0, 1], [1, 0, 0], [0, 1, 1]]
Second Matrix= [[1, 0, 1], [0, 1, 1], [1, 0, 1]]
Rows= 3 Cols= 3
Matrix Intersection [[1, 0, 1], [0, 0, 0], [0, 0, 1]]
Matrix Union [[1, 0, 1], [1, 1, 1], [1, 1, 1]]
[('p', 'p'), ('p', 'r'), ('r', 'r')]
[('p', 'p'), ('p', 'r'), ('q', 'p'), ('q', 'q'), ('q', 'r'), ('r', 'p'), ('r', 'q'), ('r', 'r')]
```

Problem #04: Write a program to find shortest path by Warshall's algorithm.

```
INF = 1000000000
def floyd_warshall(vertex, adjacency_matrix):
 # calculating all pair shortest path
 for k in range(0, vertex):
  for i in range(0, vertex):
   for j in range(0, vertex):
    # relax the distance from i to j by allowing vertex k as intermedi
ate vertex
    # consider which one is better, going through vertex k or the prev
ious value
    adjacency_matrix[i][j] = min(adjacency_matrix[i][j], adjacency_mat
rix[i][k] + adjacency_matrix[k][j])
 # pretty print the graph
 # o/d means the leftmost row is the origin vertex
 # and the topmost column as destination vertex
 print("o/d", end='')
 for i in range(0, vertex):
  print("\t{:d}".format(i+1), end='')
 print();
 for i in range(0, vertex):
  print("{:d}".format(i+1), end='')
  for j in range(0,vertex):
   print("\t{:d}".format(adjacency_matrix[i][j]), end='')
  print();
"""
input is given as adjacency matrix,
input represents this undirected graph
 A--1--B
 |    /
 3   /
 |  1
 | /
 C--2--D
should set infinite value for each pair of vertex that has no edge
 """
adjacency_matrix = [
     [  0,   5, INF, 10],
     [  INF,   0, 3, INF],
     [  INF,   INF, 0,   1],
     [INF, INF, INF,   0]
     ]
floyd_warshall(4, adjacency_matrix);
```

Output:

```
o/d  1    2    3    4
1    0    5    8    9
2    1000000000 0    3    4
3    1000000000 1000000000 0    1
4    1000000000 1000000000 1000000000 0
```

Source: https://iq.opengenus.org/floyd-warshall-algorithm-shortest-path-between-all-pair-of-nodes/

Problem #05: Write a program for the solution of graph coloring problem by Welch-Powell's algorithm.

```
def color_nodes(graph):
    color_map = {}
    # Consider nodes in descending degree
    for node in sorted(graph, key=lambda x: len(graph[x]), reverse=True):
        neighbor_colors = set(color_map.get(neigh) for neigh in graph[node])
        color_map[node] = next(
            color for color in range(len(graph)) if color not in neighbor_colors
        )
    return color_map
#Adjacent list
graph={'a':list('bcd'),'b': list('ac'),'c': list('abdef'),'d': list('ace'),'e': list('cdf'),'f': list('ce')}
print(color_nodes(graph))
```

Output:

```
{'c': 0, 'a': 1, 'd': 2, 'e': 1, 'b': 2, 'f': 2}
```