Міністерство освіти та науки України

Національний університет водного господарства та природокористування

Навчально-науковий інститут автоматики, кібернетики та обчислювальної техніки

Кафедра комп'ютерних наук та прикладної математики

# 

# **Звіт про виконання лабораторної роботи**

Виконала:

студентка групи КН-31

Войцеховська Діана

Перевірила:

к. т. н. Жуковська Н.А.

Рівне 2020

**Лабораторна робота №3. Властивості бінарних відношень (частина 2)**

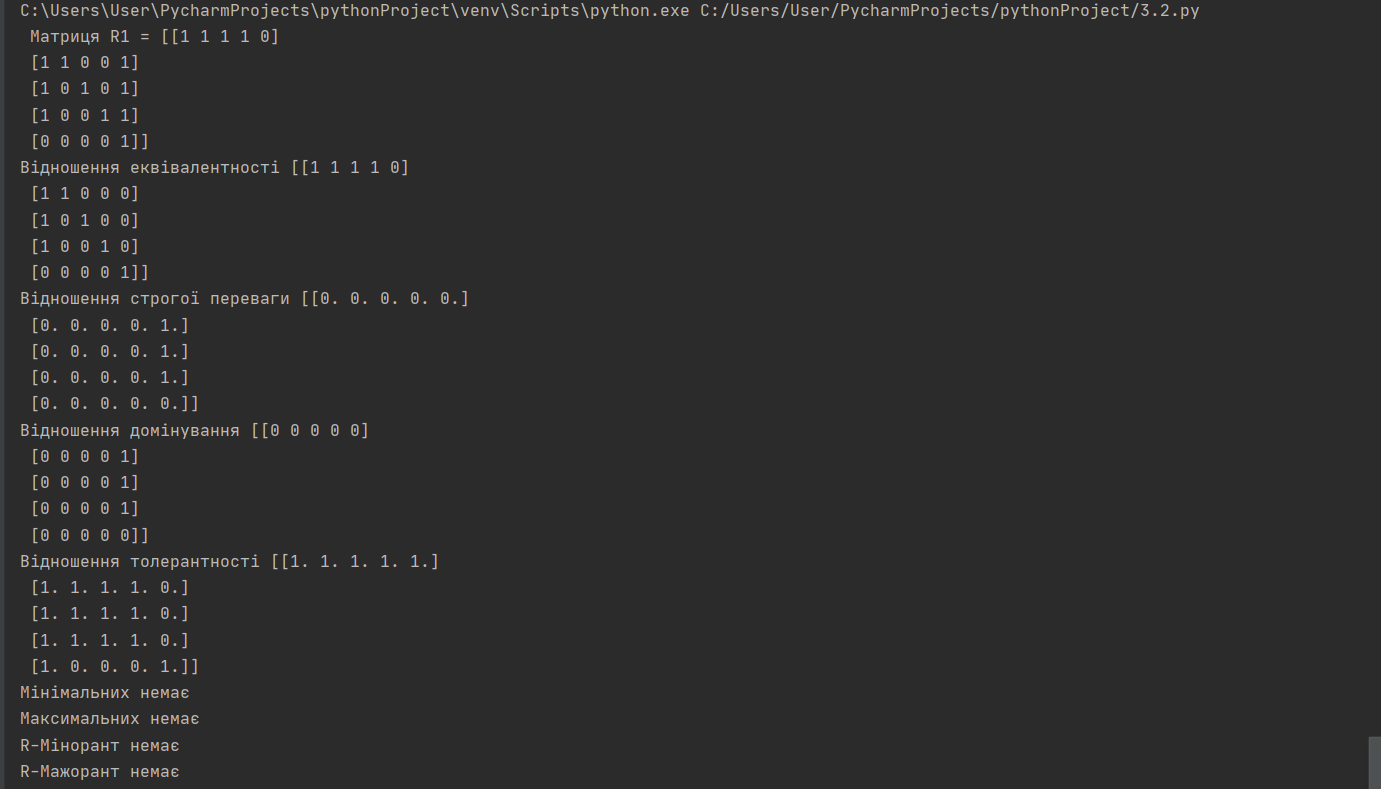
**Мета роботи:** вивчення властивостей бінарних відношень, вибір кращих альтернатив за допомогою бінарних відношень, набуття прийняття рішень на основі заданих відношень.

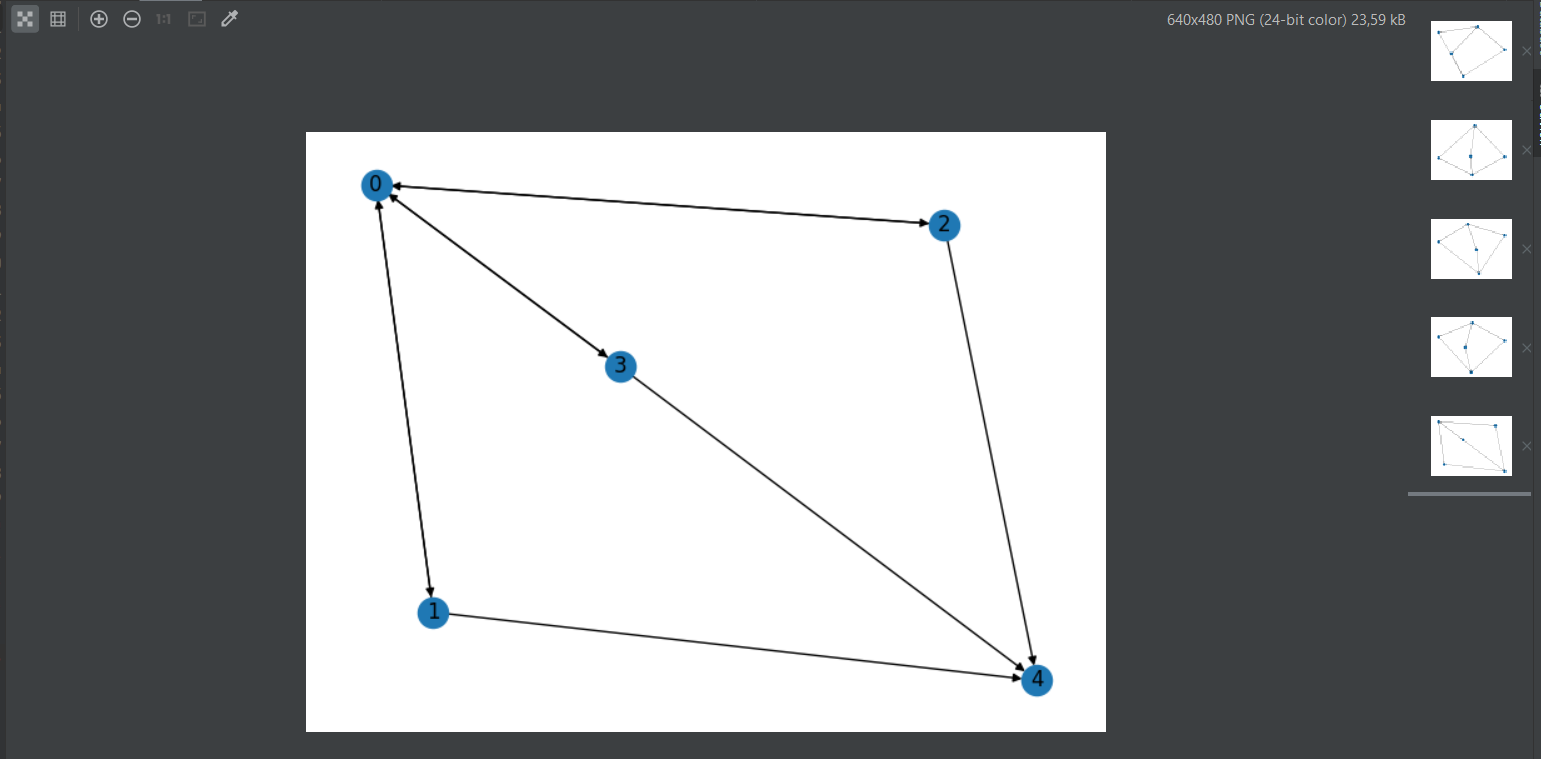
**Завдання:** відшукати для заданого згідно індивідуального варіанту бінарного відношення R найбільший, найменший, максимальний та мінімальний елементи, якщо такі існують, і побудувати відповідні даному відношенню відношення еквівалентності, домінування, толерантності та строгої переваги.

**Програмний код:**

import networkx as nx  
import matplotlib.pyplot as plt  
import numpy as np  
from numpy import linalg, bitwise\_or  
  
R1 = np.array([[1, 1, 1, 1, 0],#a  
 [1, 1, 0, 0, 1],#b  
 [1, 0, 1, 0, 1],#c  
 [1, 0, 0, 1, 1],#d  
 [0, 0, 0, 0, 1]])#e  
print(" Матриця R1 =", R1)  
  
R3 = R1.T #обернена матриця  
R4 = R1 | R3 #об'єднання матриці з оберненою  
R5 = R1 & R3 #перетин з оберненою матрицею  
print("Відношення еквівалентності", R5)  
R6 = np.zeros((5, 5))  
for i in range(0, 5):  
 for j in range(0, 5):  
 if R1[i][j] > 0 and R3[i][j] == 0:  
 R6[i][j] = 1  
 else:  
 R6[i][j] = 0  
print("Відношення строгої переваги", R6) #різниця матриці з оберненою  
R7 = 1-R1  
R8 = R7.T #обернене доповнення матрці  
print("Відношення домінування", R1 & R8) #перетин матриці з оберненим доповненням  
R9 = np.ones((5, 5))  
R10 = np.zeros((5, 5))  
for i in range(0, 5):  
 for j in range(0, 5):  
 if R9[i][j] > 0 and R4[i][j] == 0:  
 R10[i][j] = 1  
 else:  
 R10[i][j] = 0  
R12 = np.zeros((5, 5))  
for i in range(0, 5):  
 for j in range(0, 5):  
 if R10[i][j] == 1 and R5[i][j] == 0 or R10[i][j] == 0 and R5[i][j] == 1:  
 R12[i][j] = 1  
 else:  
 R12[i][j] = R10[i][j] \* R5[i][j]  
print("Відношення толерантності", R12)  
R11 = np.zeros((5, 5))  
Minimum = True  
for j in range(0, 5):  
 while Minimum:  
 if R1[i][j] == 0:  
 Minimum = False  
 print("Мінімальних немає")  
 else:  
 Minimum = True  
 break  
Maximum = True  
for i in range(0, 5):  
 while Maximum:  
 if R1[i][i] == 1:  
 Maximum = False  
 print("Максимальних немає")  
 else:  
 Maximum = True  
 break  
Minorant = True  
for i in range(0, 5):  
 while Minorant:  
 if R1[i][j] == 0:  
 Minorant = False  
 print("R-Мінорант немає")  
 else:  
 Minorant = True  
 break  
Magorant = True  
for j in range(0, 5):  
 while Magorant:  
 if R1[i][i] == 1:  
 Magorant = False  
 print("R-Мажорант немає")  
 else:  
 Magorant = True  
 break  
  
G = nx.from\_numpy\_matrix(np.matrix(R1), create\_using=nx.DiGraph)  
layout = nx.spring\_layout(G)  
nx.draw(G, with\_labels=True)  
plt.show()  
  
G1 = nx.from\_numpy\_matrix(np.matrix(R5), create\_using=nx.DiGraph)  
layout = nx.spring\_layout(G)  
nx.draw(G, with\_labels=True)  
plt.show()  
  
G2 = nx.from\_numpy\_matrix(np.matrix(R6), create\_using=nx.DiGraph)  
layout = nx.spring\_layout(G)  
nx.draw(G, with\_labels=True)  
plt.show()  
  
G3 = nx.from\_numpy\_matrix(np.matrix(R1 & R8), create\_using=nx.DiGraph)  
layout = nx.spring\_layout(G)  
nx.draw(G, with\_labels=True)  
plt.show()  
  
G3 = nx.from\_numpy\_matrix(np.matrix(R12), create\_using=nx.DiGraph)  
layout = nx.spring\_layout(G)  
nx.draw(G, with\_labels=True)  
plt.show()

**Результат роботи програми:**

****

****

**Аналіз результатів:**

В даній лабораторній роботі не виникло труднощів і всі відношення були реалізовані. Дана матриця немає ніяких максимальних, мінімальних елементів, R- мінорант, R-мажорант.