Міністерство освіти та науки України

Львівський національний університет імені Івана Франка

Факультет електроніки та

Комп’ютерних технологій

**Звіт**

Про виконання лабораторної роботи №5

“Програмна реалізація факторизації бінарного відношення”

Виконав:

Студент групи ФеІ-44

Сапанюк М.І.

Перевірила:

Мостова М. Р.

Львів 2022

**Мета:**

Засвоїти шляхи агрегування та факторизації бінарних відношень (БВ).

**Хід роботи:**

1. Опрацювати і засвоїти матеріал наведений в теоретичних відомостях.

2. Отримати від викладача матрицю відношення для роботи наприклад

3. Для отриманого відношення намалювати відповідний граф.

4. Виконати факторизацію отриманого відношення за відношенням взаємної досяжності для цього:

a. Знайти транзитивне замикання отриманого відношення;

b. Знайти відношення досяжності;

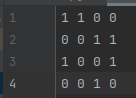
c. Знайти відношення взаємної досяжності.

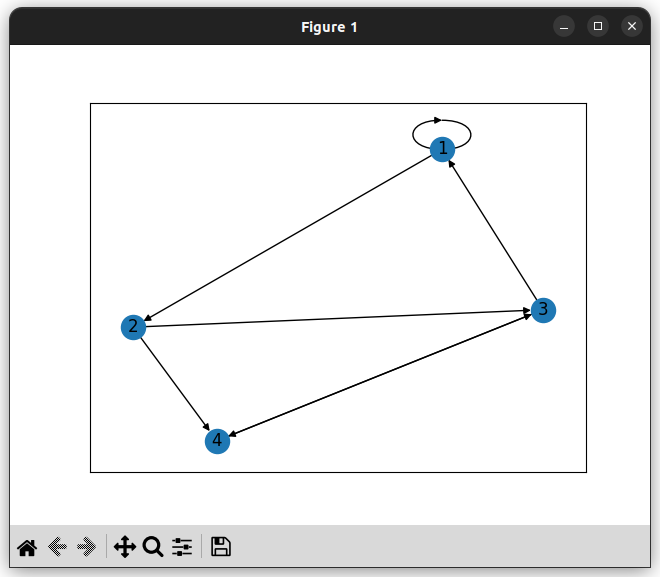
5. Намалювати граф факторизованого відношення.

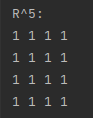
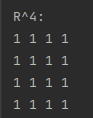
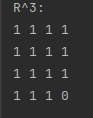
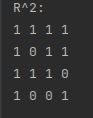
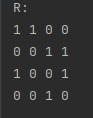
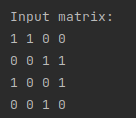
6. Написати програму, котра реалізує операції зазначенні у пункті 4, в звіті навести копії екранів з результатами роботи програми та лістинг основної (виконавчої) частини написаної програми.

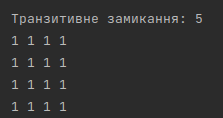
7. Зробити висновки.

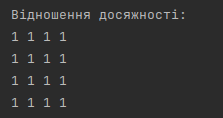
**Виконання завдання:**

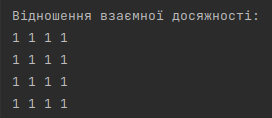


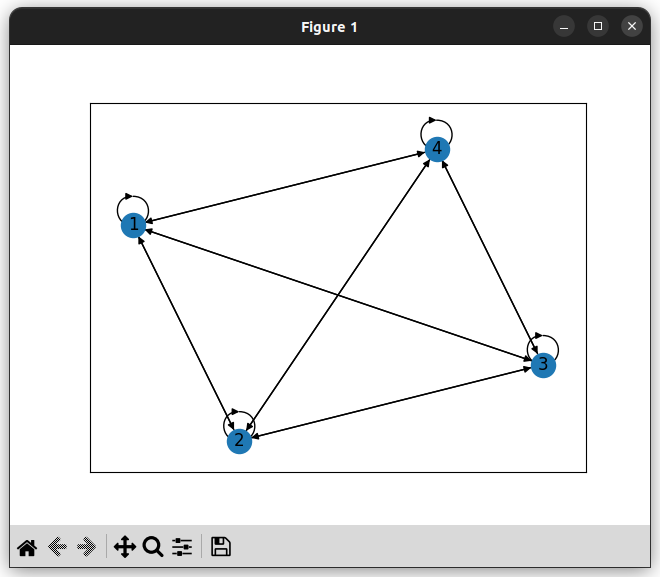












**Висновок:**

На цій лабораторній роботі я програмно реалізував факторизацію бінарного відношення.  
**Додаток:**

import os  
import networkx as nx  
import matplotlib.pyplot as plt  
  
script\_dir = os.path.dirname(\_\_file\_\_) # <-- absolute dir the script is in  
rel\_path = "matrix.txt"  
path\_to\_file = os.path.join(script\_dir, rel\_path)  
  
  
def print\_out\_matrix(matr):  
 for e in matr:  
 for e2 in e:  
 print(e2 \* 1, end=" ")  
 print()  
  
  
def intersecion(A, B):  
 lenght = len(A)  
 RES = [[0 for k in range(len(A[0]))] for n in range(lenght)]  
 for n in range(lenght):  
 for k in range(len(A[0])):  
 RES[n][k] = A[n][k] & B[n][k]  
 return RES  
  
  
def union(A, B):  
 lenght = len(A)  
 RES = [[0 for k in range(len(A[0]))] for n in range(lenght)]  
 for n in range(lenght):  
 for k in range(len(A[0])):  
 RES[n][k] = B[n][k] or A[n][k]  
 return RES  
  
  
def m(matr1, matr2):  
 matr = []  
 temp = []  
 length1 = len(matr1)  
 length2 = len(matr2)  
 for i in range(length1):  
 for j in range(len(matr2[0])):  
 summ = 0  
 for n in range(length2):  
 summ = summ or (matr1[i][n] and matr2[n][j])  
 temp.append(summ)  
 matr.append(temp)  
 temp = []  
 return matr  
  
  
def find\_transition(matr1):  
 tr = [matr1]  
 state = False  
 while (state == False):  
 matr = m(tr[-1], matr1)  
 state = equals(matr, tr[-1]) or equals(matr, matr1) or equals(matr, tr[len(tr) - 2])  
 tr.append(matr)  
 if (len(tr) > 20):  
 return []  
 return tr  
  
  
def transition(tr):  
 matr = tr[0]  
 for i in range(1, len(tr)):  
 matr = union(matr, tr[i])  
 return matr  
  
  
def equals(matr1, matr2):  
 length1 = len(matr1)  
 for i in range(length1):  
 for j in range(len(matr2[0])):  
 if matr1[i][j] != matr2[i][j]:  
 return False  
 return True  
  
  
def reaching(matr1):  
 A = [[True if n == k else False for k in range(len(matr1[0]))] for n in range(len(matr1))]  
 return union(A, matr1)  
  
  
def reaching\_vsaem(matr1):  
 return intersecion(matr1, transpose\_marix(matr1))  
  
  
def transpose\_marix(matr1):  
 length1 = len(matr1)  
 matr = [[0 for k in range(len(matr1[0]))] for n in range(length1)]  
 for n in range(length1):  
 for k in range(len(matr1[0])):  
 matr[k][n] = matr1[n][k]  
 return matr  
  
  
def draw(mat):  
 G = nx.DiGraph()  
 G.add\_nodes\_from(range(len(mat), 1))  
 res = []  
 for indexRow in range(len(mat)):  
 for indexCol in range(len(mat[0])):  
 if mat[indexRow][indexCol] == 1:  
 res.append((indexRow + 1, indexCol + 1))  
 G.add\_edges\_from(res)  
 nx.draw\_networkx(G)  
 plt.show()  
  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 with open(path\_to\_file, 'r') as file:  
 matr = [list(map(lambda x: True if x == '1' else False, line.replace("\n", '').split())) for line in  
 file.readlines()]  
 print("Input matrix:")  
 print\_out\_matrix(matr)  
 temp = find\_transition(matr)  
 for k in range(len(temp)):  
 if (k == 0):  
 print("\nR:")  
 else:  
 print("R^" + str(k + 1) + ": ")  
 print\_out\_matrix(temp[k])  
 print()  
  
 print("\nТранзитивне замикання: " + str(len(temp)))  
 transition = transition(temp)  
 print\_out\_matrix(transition)  
  
 print("\nВідношення досяжності:")  
 dosisgnosti = reaching(transition)  
 print\_out\_matrix(dosisgnosti)  
  
 print("\nВідношення взаємної досяжності:")  
 print\_out\_matrix(reaching\_vsaem(dosisgnosti))  
  
 draw(matr)  
 draw(reaching\_vsaem(dosisgnosti))