Міністерство освіти та науки України

Львівський національний університет імені Івана Франка

Факультет електроніки та

Комп’ютерних технологій

**Звіт**

Про виконання лабораторної роботи №5

“ Аналіз ефективності комунікації на основі транзитивного замикання нечітких бінарних відношень”

Виконав:

Студент групи ФеІ-44

Сапанюк М.І.

Перевірила:

Притула М.

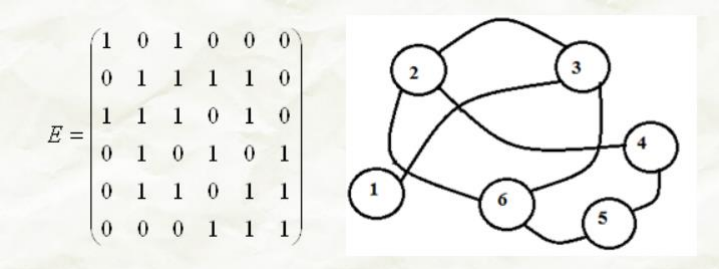
Львів 2022

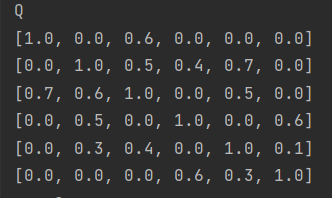
**Мета:**

ознайомитися з основними властивостями бінарних нечітких відношень, застосувати транзитивне замикання бінарних нечітких відношень у практичних задачах.

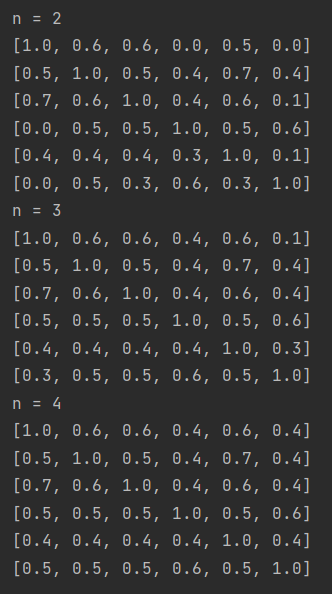
**Хід роботи:**

1. Розглянути задачу про ефективність комунікації у певній сукупності людей, наприклад, академічній групі студентів. Бінарне нечітке відношення Q = {, µQ()}, задане на декартовому добутку універсумів Х × Х, де Х = {х1, x2, …, хn} – сукупність людей, описує умову “особа хi добре знайома з xj” (очевидно, що µQ() = 1). Бінарне нечітке відношення Q є рефлексивним та симетричним. Його ще називають відношенням толерантності або нечіткої подібності.   
2. Задати бінарне нечітке відношення Q, яке описує умову “особа хi добре знайома з xj” в академічній групі студентів.





3. Запрограмувати алгоритм, який на основі заданого бінарного нечіткого відношення Q за допомогою операції транзитивного замикання знайде матрицю нечіткого відношення, що описує ефективність поширення інформації.



4. Розглянути задачу про ефективність комунікації в іншій сукупності людей. Порівняти результати.

**Висновок:**

Виконавши лабораторну роботу я, ознайомився з основними властивостями бінарних нечітких відношень, застосував транзитивне замикання бінарних нечітких відношень на практичній задачі

**Додаток:**

from typing import List  
  
  
def printMatrix(matrix: List[List[float]]) -> None:  
 for row in matrix:  
 print(row)  
  
  
def composition(matrix1: List[List[float]], matrix2: List[List[float]]):  
 answer: List[List[float]] = []  
 size = len(matrix1)  
 for i in range(0, size):  
 answer.append([])  
 for j in range(0, size):  
 temp: List[float] = []  
 for k in range(0, size):  
 temp.append(min(matrix1[i][k], matrix2[k][j]))  
 answer[i].append(float(format(max(temp), '.2f')))  
 return answer  
  
  
def tranzit\_lock(matrix: List[List[float]]):  
 prev\_matrix: List[List[float]] = [[]]  
 current\_matrix: List[List[float]] = [[]]  
 prev\_matrix = matrix  
 n = 2  
 while 1:  
 current\_matrix = composition(matrix, prev\_matrix)  
 if current\_matrix == prev\_matrix:  
 break  
 prev\_matrix = current\_matrix  
 print(f'n = {n}')  
 printMatrix(prev\_matrix)  
 n += 1  
  
  
def main() -> int:  
 Q: List[List[float]] = [[1.0, 0.0, 0.6, 0.0, 0.0, 0.0],  
 [0.0, 1.0, 0.5, 0.4, 0.7, 0.0],  
 [0.7, 0.6, 1.0, 0.0, 0.5, 0.0],  
 [0.0, 0.5, 0.0, 1.0, 0.0, 0.6],  
 [0.0, 0.3, 0.4, 0.0, 1.0, 0.1],  
 [0.0, 0.0, 0.0, 0.6, 0.3, 1.0]]  
 # Q: list[list[float]] = [[1.0, 0.0, 1.0, 0.0, 0.0, 0.0],  
 # [0.0, 1.0, 1.0, 1.0, 1.0, 0.0],  
 # [1.0, 1.0, 1.0, 0.0, 1.0, 0.0],  
 # [0.0, 1.0, 0.0, 1.0, 0.0, 1.0],  
 # [0.0, 1.0, 1.0, 0.0, 1.0, 1.0],  
 # [0.0, 0.0, 0.0, 1.0, 1.0, 1.0]]  
 print('Q')  
 printMatrix(Q)  
 tranzit\_lock(Q)  
 return 0  
  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 main()