Міністерство освіти та науки України
Львівський національний університет імені Івана Франка

Факультет електроніки та
Комп’ютерних технологій

Звіт

Про виконання лабораторної роботи №3
“Операції над бінарними відношеннями та їх властивості”

Виконав:
Студент групи ФеІ-44
Сапанюк М.І.
Перевірив:
Мостова М.Р.

Львів 2022

Мета:

Побудова графу бінарного відношення. Знаходження перетину, обєднання, композиції бінарних відношень.

Хід роботи:

Завдання: 1.Написати програму, яка зображає граф бінарного відношення, якщо бінарне відношення записане у матричному вигляді.

Завдання 3 Написати програму, яка знаходить перетин, об'єднання, композиції двох бінарних відношень. При заданні бінарних відношень здійснити перевірку, чи вказані операції є допустимими. Для результуючих бінарних відношень знайти та вивести у програмі відношення доповнення. обернене відношення.

Виконання завдання:

```
def printMatrix(mat):
    for row in mat:
        print(row)


def printGraph(A, mat):
    print('Граф: ')
    string = ''
    for rowIndex in range(len(mat)):
        string += f"{A[rowIndex]} -> "
        for colIndex in range(len(mat[rowIndex])):
            if mat[rowIndex][colIndex] == 1:
                string += f"{A[colIndex]} "
        print(string)
        string = ''


def intersection(mat1, mat2):
    result = [[0 for x in range(len(mat1))] for y in range(len(mat1[0]))]
    for indexRow in range(len(mat1)):
        for indexCol in range(len(mat1[indexRow])):
            result[indexRow][indexCol] = mat1[indexRow][indexCol] & mat2[indexRow][indexCol]
    return result


def union(mat1, mat2):
    result = [[0 for x in range(len(mat1))] for y in range(len(mat1[0]))]
    for indexRow in range(len(mat1)):
        for indexCol in range(len(mat1[indexRow])):
            result[indexRow][indexCol] = mat1[indexRow][indexCol] | mat2[indexRow][indexCol]
    return result


def composition(mat1, mat2):
    result = [[0 for x in range(len(mat1))] for y in range(len(mat1[0]))]
    for indexRow in range(len(mat1)):
        for indexCol in range(len(mat1[indexRow])):
            for i in range(len(mat1[indexRow])):
                result[indexRow][indexCol] += (mat1[indexRow][i] * mat2[i][indexCol])
            if result[indexRow][indexCol] > 0: result[indexRow][indexCol] = 1
    return result


def addition(mat):
```

```
    result = [[0 for x in range(len(mat))] for y in range(len(mat[0]))]
    for indexRow in range(len(mat)):
        for indexCol in range(len(mat)):
            result[indexRow][indexCol] = 1 if mat[indexRow][indexCol] == 0 else 0
    return result


def reverse(mat):
    result = [[0 for x in range(len(mat))] for y in range(len(mat[0]))]
    for indexRow in range(len(mat)):
        for indexCol in range(len(mat[indexRow])):
            result[indexRow][indexCol] = mat[indexCol][indexRow]
    return result


def isReflexive(mat):
    for index in range(len(mat)):
        if mat[index][index] != 1:
            return False
    return True


def isSymmetrical(mat):
    for indexRow in range(len(mat)):
        for indexCol in range(len(mat)):
            if mat[indexRow][indexCol] != mat[indexCol][indexRow]:
                return False
    return True


def isTransitive(mat):
    c = composition(mat, mat)
    for indexRow in range(len(mat)):
        for indexCol in range(len(mat)):
            if mat[indexRow][indexCol] == 1:
                if c[indexRow][indexCol] == 1:
                    continue
                else:
                    return False
    return True




P = [
    [1, 0, 0, 0],
    [1, 1, 1, 1],
    [1, 0, 1, 1],
    [1, 0, 1, 1],
]

Q = [
    [1, 0, 0, 0],
    [0, 1, 0, 0],
    [0, 0, 1, 1],
    [0, 0, 1, 1],
]

A = ["З", "Л", "В", "О"]
printGraph(A, P)
print()

ref, sym, tra = isReflexive(P), isSymmetrical(P), isTransitive(P)
print("Рефлексивність - ", ref)
print("Симетричність - ", sym)
```

```
print("Транзитивність - ", tra)
print("Чи є воно відношенням еквівалентності - ", ref and sym and tra)
print()

print("Перетин:")
i = intersection(P, Q)
printMatrix(i)
print("Доповнення:")
printMatrix(addition(i))
print("Обернене відношення:")
printMatrix(reverse(i))
print()

print("Об'єднання:")
u = union(P, Q)
printMatrix(u)
print("Доповнення:")
printMatrix(addition(u))
print("Обернене відношення:")
printMatrix(reverse(u))
print()

print("Композиція:")
c = composition(P, Q)
printMatrix(c)
print("Доповнення:")
printMatrix(addition(c))
print("Обернене відношення:")
printMatrix(reverse(c))
```

Результат виконання:

```
Граф:
З -> З
Л -> З Л В О
В -> З В О
О -> З В О

Рефлексивність -  True
Симетричність -  False
Транзитивність -  True
Чи є воно відношенням еквівалентності -  False
```

```
Перетин:
[1, 0, 0, 0]
[0, 1, 0, 0]
[0, 0, 1, 1]
[0, 0, 1, 1]
Доповнення:
[0, 1, 1, 1]
[1, 0, 1, 1]
[1, 1, 0, 0]
[1, 1, 0, 0]
Обернене відношення:
[1, 0, 0, 0]
[0, 1, 0, 0]
[0, 0, 1, 1]
[0, 0, 1, 1]

Об'єднання:
[1, 0, 0, 0]
[1, 1, 1, 1]
[1, 0, 1, 1]
[1, 0, 1, 1]
Доповнення:
[0, 1, 1, 1]
[0, 0, 0, 0]
[0, 1, 0, 0]
[0, 1, 0, 0]
Обернене відношення:
[1, 1, 1, 1]
[0, 1, 0, 0]
[0, 1, 1, 1]
[0, 1, 1, 1]

Композиція:
[1, 0, 0, 0]
[1, 1, 1, 1]
[1, 0, 1, 1]
[1, 0, 1, 1]
Доповнення:
[0, 1, 1, 1]
[0, 0, 0, 0]
[0, 1, 0, 0]
[0, 1, 0, 0]
Обернене відношення:
[1, 1, 1, 1]
[0, 1, 0, 0]
[0, 1, 1, 1]
[0, 1, 1, 1]
```

Висновок:

На цій лабораторній роботі я навчився будувати граф бінарного відношення та знаходити перетин, об’єднання, композицію бінарних відношень.