



Министерство науки и высшего образования Российской Федерации
Калужский филиал
федерального государственного бюджетного
образовательного учреждения высшего образования
«Московский государственный технический университет
имени Н.Э. Баумана
(национальный исследовательский университет)»
(КФ МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ

ИУК «Информатика и управление»

КАФЕДРА

**ИУК4 «Программное обеспечение ЭВМ,
информационные технологии»**

ЛАБОРАТОРНАЯ РАБОТА №2

«Замыкание отношений»

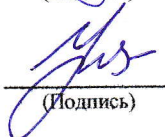
ДИСЦИПЛИНА: «Дискретная математика»

Выполнил: студент гр.ИУК4-32Б


(Подпись)

(Зудин Д. В.)
(Ф.И.О.)

Проверил:


(Подпись)

(Никитенко У. В.)
(Ф.И.О.)

Дата сдачи (защиты): 24.10.2022

Результаты сдачи (защиты):

- Бальная оценка: 70

- Оценка: зачтено

Калуга, 2022 г.

Цели исследование свойств бинарных отношений, построение замыкания отношений

Задачи. определить какими свойствами обладают бинарные отношения, построение композиции отношений, построение рефлексивного, симметричного и транзитивного замыкания отношений

ЗАДАНИЕ

Написать программу, в которой: Дано число n и два бинарных отношения на множестве размера n . Для каждого из этих отношений определяется, являются ли они рефлексивными, антирефлексивными, симметричными, антисимметричными и транзитивными, а так же найдите их композицию. Найдите рефлексивное, симметричное транзитивное (двумя способами: с помощью умножения и сложения матриц; с помощью алгоритма Уоршалла) замыкание.

Этапы выполнения программы:

1. Вывод основного меню

```
C:\pythonProject\venv\Scripts\python.exe C:\pythonProject\main.py
Выберите пункт меню
1. Создать бинарные отношения
2. Проверка отношений на рефлексивность
3. Проверка отношений на антирефлексивность
4. Проверка отношений на симметричность
5. Проверка отношений на антисимметричность
6. Проверка отношений на транзитивность
7. Композиция отношений
8. Рефлексивное замыкание
9. Транзитивное замыкание (Метод умножения и сложения матриц)
10. Транзитивное замыкание (Метод Уоршалла)
0. Выход
Введите номер пункта:
```

2. Создание бинарных отношений

```
Введите номер пункта: 1
Введите мощность множества (1 <= n <= 100): 3
Введите 1-й элемент 1-го бинарного отношения (R1): 1 0 0
Введите 2-й элемент 1-го бинарного отношения (R1): 1 0 1
Введите 3-й элемент 1-го бинарного отношения (R1): 1 1 1

Введите 1-й элемент 2-го бинарного отношения (R2): 0 1 1
Введите 2-й элемент 2-го бинарного отношения (R2): 0 0 0
Введите 3-й элемент 2-го бинарного отношения (R2): 0 1 1
R1:
1 0 0
1 0 1
1 1 1
R2:
0 1 1
0 0 0
0 1 1
Чтобы вернуться назад нажмите "z"
```

2. Проверка отношений на рефлексивность

```
Введите номер пункта: 2
R1:
1 0 0
1 0 1
1 1 1
R2:
0 1 1
0 0 0
0 1 1
(R1) Не Симметричность (False) - 0
(R2) Не Симметричность (False) - 0
Чтобы вернуться назад нажмите "z"
```

3. Проверка отношений на антирефлексивность

```
Введите номер пункта: 3
R1:
1 0 0
1 0 1
1 1 1
R2:
0 1 1
0 0 0
0 1 1
(R1) Не antiСимметричность (False) - 0
(R2) Не antiСимметричность (False) - 0
Чтобы вернуться назад нажмите "z"
```

4. Проверка отношений на симметричность

```
Введите номер пункта: 4
R1:
1 0 0
1 0 1
1 1 1
R2:
0 1 1
0 0 0
0 1 1
(R1) Не simmetry (False) - 0
(R2) Не simmetry (False) - 0
Чтобы вернуться назад нажмите "z"
```

5. Проверка отношений на антисимметричность

```

Введите номер пункта: 5
R1:
1 0 0
1 0 1
1 1 1
R2:
0 1 1
0 0 0
0 1 1
(R1) Ne antiSymmetry (False) - 0
(R2) AntiSymmetry (True) - 1
Чтобы вернуться назад нажмите "z"

```

6. Проверка отношений на транзитивность

```

Введите номер пункта: 6
R1:
1 0 0
1 0 1
1 1 1
R2:
0 1 1
0 0 0
0 1 1
Транзитивная матрица tr:
1 0 0
1 1 1
1 1 1
(R1) Ne transitivity (False) - 0
Транзитивная матрица tr:
0 1 1
0 0 0
0 1 1
(R2) Транзитивность (True) - 1
Чтобы вернуться назад нажмите "z"

```

7. Композиция отношений

```
Введите номер пункта: 7
R1:
1 0 0
1 0 1
1 1 1
R2:
0 1 1
0 0 0
0 1 1
Композиция R1 and R2:
0 1 1
0 1 1
0 1 1
Чтобы вернуться назад нажмите "z"
```

8. Рефлексивное замыкание

```
Введите номер пункта: 8
R1:
1 0 0
1 0 1
1 1 1
R2:
0 1 1
0 0 0
0 1 1
Бинарное отношение R1:
Рефлексивные замыкания, которые есть:
[(0, 0), (2, 2)]
Рефлексивные замыкания, которых нет:
[(1, 1)]
Бинарное отношение R2:
Рефлексивные замыкания, которые есть:
[(2, 2)]
Рефлексивные замыкания, которых нет:
[(0, 0), (1, 1)]
Чтобы вернуться назад нажмите "z"
```

9. Транзитивное замыкание (Метод умножения и сложения матриц)

Введите номер пункта: 9

R1:

1 0 0

1 0 1

1 1 1

R2:

0 1 1

0 0 0

0 1 1

Транзитивное замыкание для R1 (Методом умножения и сложения матриц):

Матрица tr_1:

1 0 0

1 1 1

1 1 1

Матрица tr_2:

1 0 0

1 1 1

1 1 1

Матрица tr:

1 0 0

1 1 1

1 1 1

Транзитивное замыкание для R2 (Методом умножения и сложения матриц):

Матрица tr_1:

0 1 1

0 0 0

0 1 1

Матрица tr_2:

0 1 1

0 0 0

0 1 1

Матрица tr:

0 1 1

0 0 0

0 1 1

Чтобы вернуться назад нажмите "z"

10. Транзитивное замыкание (Метод Уоршалла)

```

Введите номер пункта: 10
R1:
1 0 0
1 0 1
1 1 1
R2:
0 1 1
0 0 0
0 1 1
Транзитивное замыкание для R1 (Методом Уоршалла):
1 1 1
1 1 1
1 1 1
Транзитивное замыкание для R2 (Методом Уоршалла):
0 1 1
0 1 1
0 1 1
Чтобы вернуться назад нажмите "z"
|

```

0. Выход

```

Введите номер пункта: 0
Выход из программы!

Process finished with exit code 0

```

Листинг программы:

```

import os

def clear():
    os.system("clear")

def ПечатьОтношений(m, n):
    print("R1:")
    for line in m:
        print(*line)

    print("R2:")
    for line in n:
        print(*line)

    for i in range(len(m)):
        if m[i][i] != 1:
            return False
    return True

def Симметричность(m):
    for i in range(len(m)):
        if m[i][i] != 1:
            return False

```



```

return True

def АнтиСимметричность(m):
    for i in range(len(m)):
        if m[i][i] != 0:
            return False
    return True

def Симетр(m):
    for i in range(len(m)):
        for j in range(len(m)):
            if i != j:
                if m[i][j] != m[j][i]:
                    return False
    return True

def АнтиСиметр(m):
    n = len(m)
    nm = [[0 for _ in range(n)] for _ in range(n)]
    for i in range(n):
        for j in range(n):
            nm[i][j] = m[i][j] * m[j][i]

    flag = True
    for i in range(len(m)):
        for j in range(len(m)):
            if i != j:
                if nm[i][j] != 0:
                    flag = False
    return flag

def Транзитивность(m):
    length = len(m)
    tr = [[0 for _ in range(0, length)] for _ in range(0, length)]
    flag = True

    for i in range(len(m)):
        for j in range(len(m)):
            for k in range(len(m)):
                tr[i][j] = (tr[i][j] | (m[i][k] & m[k][j]))

    print("Транзитивная матрица tr:")
    for line in tr:
        print(*line)

    if (m != tr):
        flag = False
    return flag

def Композиция(m, n):
    length = len(m)
    comp = [[0 for _ in range(0, length)] for _ in range(0, length)]

    for i in range(len(m)):
        for j in range(len(m)):
            for k in range(len(m)):
                comp[i][j] = comp[i][j] + m[i][k] * n[k][j]
            if comp[i][j] != 0:
                comp[i][j] = 1

```

```

print("Композиция R1 and R2:")
for line in comp:
    print(*line)

def РефлексивныеЗамыкания(m):
    print("Рефлексивные замыкания, которые есть:")
    print([(i, i) for i in range(len(m)) if m[i][i] == 1])
    print("Рефлексивные замыкания, которых нет:")
    print([(i, i) for i in range(len(m)) if m[i][i] == 0])

def ТранзитивныеЗамыкания1(m):
    length = len(m)
    tr_1 = [[0 for _ in range(0, length)] for _ in range(0, length)]

    for i in range(len(m)):
        for j in range(len(m)):
            for k in range(len(m)):
                tr_1[i][j] = (tr_1[i][j] | (m[i][k] & m[k][j]))

    print("Матрица tr_1:")
    for line in tr_1:
        print(*line)

    tr_2 = [[0 for _ in range(0, length)] for _ in range(0, length)]

    for i in range(len(m)):
        for j in range(len(m)):
            tr_2[i][j] = m[i][j] | tr_1[i][j]

    print("Матрица tr_2:")
    for line in tr_2:
        print(*line)

    tr = [[0 for _ in range(0, length)] for _ in range(0, length)]

    for i in range(len(m)):
        for j in range(len(m)):
            for k in range(len(m)):
                tr[i][j] = (tr[i][j] | (tr_2[i][k] & tr_2[k][j]))

    print("Матрица tr:")
    for line in tr:
        print(*line)

def ТранзитивныеЗамыкания2(m):
    def sum_string(m, a, b):
        n = len(m)
        for i in range(n):
            m[a][i] |= m[b][i]
        return m

    length = len(m)

    pre_arr = None
    max_count = 3
    count = 0
    while True:
        for i in range(length):
            for j in range(length):
                if (i != j):

```

```

        m = sum_string(m, i, j)
        if m == pre_arr:
            count += 1
        else:
            count = 0
        pre_arr = [a[:] for a in m]
        if count >= max_count:
            return m

```

```

def пункт1():
    clear()

    global n
    n = int(input('Введите мощность множества (1 <= n <= 100): '))

    global R1
    R1 = []
    for i in range(n):
        print('Введите ' + str(i + 1) + '-й элемент 1-го бинарного отношения (R1): ',
end='')
        R1.append(list(map(int, input().split())))

    print()
    global R2
    R2 = []
    for i in range(n):
        print('Введите ' + str(i + 1) + '-й элемент 2-го бинарного отношения (R2): ',
end='')
        R2.append(list(map(int, input().split())))

    ПечатьОтношений(R1, R2)

    вназад()

```

```

def пункт2():
    clear()

    ПечатьОтношений(R1, R2)

    if (Симметричность(R1)):
        print("(R1) Симметричность (True) - 1")
    else:
        print("(R1) Не Симметричность (False) - 0")

    if (Симметричность(R2)):
        print("(R2) Симметричность (True) - 1")
    else:
        print("(R2) Не Симметричность (False) - 0")

    вназад()

```

```

def пункт3():
    clear()

    ПечатьОтношений(R1, R2)

    if (АнтиСимметричность(R1)):
        print("(R1) АнтиСимметричность (True) - 1")
    else:
        print("(R1) Не antiСимметричность (False) - 0")

```

```

if (АнтиСимметричность(R2)):
    print("(R2) antiСимметричность (True) - 1")
else:
    print("(R2) Ne antiСимметричность (False) - 0")

вназад()

def пункт4():
    clear()

    ПечатьОтношений(R1, R2)

    if (Симетр(R1)):
        print("(R1) Simmetry (True) - 1")
    else:
        print("(R1) Ne simmetry (False) - 0")

    if (Симетр(R2)):
        print("(R2) Simmetry (True) - 1")
    else:
        print("(R2) Ne simmetry (False) - 0")

    вназад()

def пункт5():
    clear()

    ПечатьОтношений(R1, R2)

    if (АнтиСиметр(R1)):
        print("(R1) AntiSimmetry (True) - 1")
    else:
        print("(R1) Ne antiSimmetry (False) - 0")

    if (АнтиСиметр(R2)):
        print("(R2) AntiSimmetry (True) - 1")
    else:
        print("(R2) Ne antiSimmetry (False) - 0")

    вназад()

def пункт6():
    clear()

    ПечатьОтношений(R1, R2)

    if (Транзитивность(R1)):
        print("(R1) Транзитивность (True) - 1")
    else:
        print("(R1) Ne transitivity (False) - 0")

    if (Транзитивность(R2)):
        print("(R2) Транзитивность (True) - 1")
    else:
        print("(R2) Ne transitivity (False) - 0")

    вназад()

def пункт7():

```

```

clear()

ПечатьОтношений(R1, R2)

Композиция(R1, R2)

вназад()

def пункт8():
    clear()

    ПечатьОтношений(R1, R2)

    print("Бинарное отношение R1:")
    РефлексивныеЗамыкания(R1)
    print("Бинарное отношение R2:")
    РефлексивныеЗамыкания(R2)

    vissza()

def пункт9():
    clear()

    ПечатьОтношений(R1, R2)

    print("Транзитивное замыкание для R1 (Методом умножения и сложения матриц):")
    ТранзитивныеЗамыкания1(R1)
    print("Транзитивное замыкание для R2 (Методом умножения и сложения матриц):")
    ТранзитивныеЗамыкания1(R2)

    vissza()

def пункт10():
    clear()

    ПечатьОтношений(R1, R2)

    print("Транзитивное замыкание для R1 (Методом Уоршола):")
    ТранзитивныеЗамыкания2(R1)
    for line in R1:
        print(*line)
    print("Транзитивное замыкание для R2 (Методом Уоршола):")
    ТранзитивныеЗамыкания2(R2)
    for line in R2:
        print(*line)

    vissza()

def menu():
    print('Выберите пункт меню')
    print(
        '1. Создать бинарные отношения\n' + '2. Проверка отношений на\n' +
        'рефлексивность\n' + '3. Проверка отношений на антирефлексивность\n' +
        '4. Проверка отношений на симметричность\n' + '5. Проверка отношений на\n' +
        'антисимметричность\n' + '6. Проверка отношений на транзитивность\n' +
        '7. Композиция отношений\n' + '8. Рефлексивное замыкание\n' + '9.\n' +
        'Транзитивное замыкание (Метод умножения и сложения матриц)\n' +
        '10. Транзитивное замыкание (Метод Уоршола)\n' + '0. Выход')

    command = int(input('Введите номер пункта: '))

```

```

if command == 1:
    пункт1()
elif command == 2:
    пункт2()
elif command == 3:
    пункт3()
elif command == 4:
    пункт4()
elif command == 5:
    пункт5()
elif command == 6:
    пункт6()
elif command == 7:
    пункт7()
elif command == 8:
    пункт8()
elif command == 9:
    пункт9()
elif command == 10:
    пункт10()
elif command == 0:
    print('Выход из программы!')
    exit(0)
else:
    print('Неверная команда')

def вназад():
    choise = input('Чтобы вернуться назад нажмите "z"\n')
    if choise == 'z':
        clear()
        menu()
    else:
        print('Неверная команда')
        вназад()

if __name__ == '__main__':
    menu()

```

Вывод: в ходе выполнения лабораторной работы были исследованы свойства бинарных отношений ,а также написана программа для определения свойств заданных бинарных отношений, построения композиции и нахождения замыкания.