Министерство образования Республики Беларусь

Учреждение образования

«Брестский государственный технический университет»

Кафедра ИИТ

Лабораторная работа №2

за 1 семестр

По дисциплине: «Дискретная математика»

Тема: «Отношения. Функции»

Выполнил:

Студент 2 курса

Группы ПО-4(1)

Иваненко И.Л.

Проверил:

Глущенко Т.А.

2020

Лабораторная работа №2

Отношения. Функции

Цель работы: изучить базовые понятия о свойствах отношений и функций

Вариант 10

***Отношения***

Задание 1

На множестве *A = {1,2,3,4,5,6,7,8,9,10}* заданы отношения  и  согласно вашему варианту. Варианты заданий указаны в *таблице 1*.

|  |  |  |
| --- | --- | --- |
| *10.* |  |  |

1. Для заданных отношений составить *матрицы отношений*. Построить орграфы отношений.

Код программы:

Модуль funcs.py:

import itertools

import numpy

import math

# ФУНКЦИЯ ВОЗВРАЩАЕТ ДЕКАРТОВО ПРОИЗВЕДЕНИЯ МНОЖЕСТВА

def get\_decart\_mult(array):

decart\_mult = []

for i in itertools.product(array, array):

decart\_mult.append(i)

return decart\_mult

# ФУНКЦИЯ ВОЗВРАЩАЕТ TRUE, ЕСЛИ ПАРА ДЕКАРТОВА ПРОИЗВЕДЕНИЯ

# ОТНОСИТСЯ К ОТНОШЕНИЮ R1

def R1\_build(pair):

return pair[0] % 4 == pair[1] % 4

# ФУНКЦИЯ ВОЗВРАЩАЕТ TRUE, ЕСЛИ ПАРА ДЕКАРТОВА ПРОИЗВЕДЕНИЯ

# ОТНОСИТСЯ К ОТНОШЕНИЮ R1

def R2\_build(pair):

return pair[0] + pair[1] == 8

# ФУНКЦИЯ ВОЗВРАЩАЕТ ОТНОШЕНИЕ R1

def get\_R1(array):

decart\_mult = get\_decart\_mult(array)

R1 = []

for pair in decart\_mult:

if R1\_build(pair) == True:

R1.append(pair)

return R1

# ФУНКЦИЯ ВОЗВРАЩАЕТ ОТНОШЕНИЕ R2

def get\_R2(array):

decart\_mult = get\_decart\_mult(array)

R2 = []

for pair in decart\_mult:

if R2\_build(pair) == True:

R2.append(pair)

return R2

# ФУНКЦИЯ ВОЗВРАЩАЕТ МАТРИЦУ ОТНОШЕНИЙ

# ФУНКЦИЯ ВОЗВРАЩАЕТ МАТРИЦУ ОТНОШЕНИЙ

def matrix\_build(R, array):

matrix = numpy.zeros((len(array), len(array)), dtype=numpy.int)

for pair in R:

matrix[pair[0] - 1][pair[1] - 1] = 1

return matrix

Модуль main:

#ЗАДАНИЕ 1

import funcs

A = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

#ЗАДАНИЕ 1

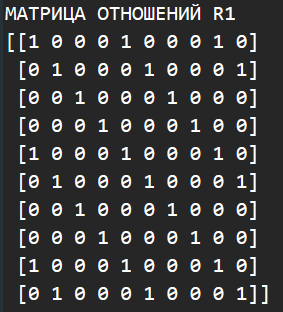
print("МАТРИЦА ОТНОШЕНИЙ R1")

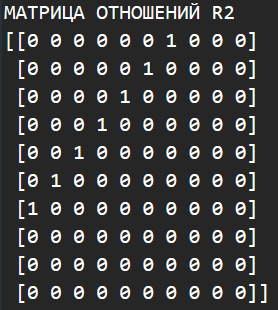
print(funcs.matrix\_build(R = funcs.get\_R1(A), array = A))

print("\n\n\nМАТРИЦА ОТНОШЕНИЙ R2")

print(funcs.matrix\_build(R = funcs.get\_R2(A), array = A))

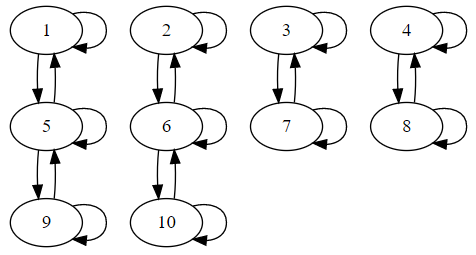
Результат выполнения:



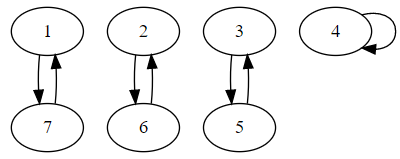


Орграфы отношений:

R1:



R2:



2. Найти обратные отношения и дополнения отношений:

Код программы:

Модуль funcs:

# ФУНКЦИЯ ВОЗВРАЩАЕТ ОБРАТНОЕ ОТНОШЕНИЕ

def get\_inverse\_relation(R):

inverse\_relation = []

for pair in R:

temp\_pair = pair

pair = []

pair.append(temp\_pair[1])

pair.append(temp\_pair[0])

inverse\_relation.append(pair)

return inverse\_relation

# ФУНКЦИЯ ВОЗВРАЩАЕТ ДОПОЛНЕНИЕ ОТНОШЕНИЯ

def get\_relation\_addition(array, R):

decart\_mult = get\_decart\_mult(array)

relation\_addition = []

for pair in decart\_mult:

if pair not in R:

relation\_addition.append(pair)

return relation\_addition

Модуль main:

#ЗАДАНИЕ 2

print("ОБРАТНОЕ ОТНОШЕНИЕ R1")

print(funcs.get\_inverse\_relation(R = funcs.get\_R1(A)))

print("ОБРАТНОЕ ОТНОШЕНИЕ R2")

print(funcs.get\_inverse\_relation(R = funcs.get\_R2(A)))

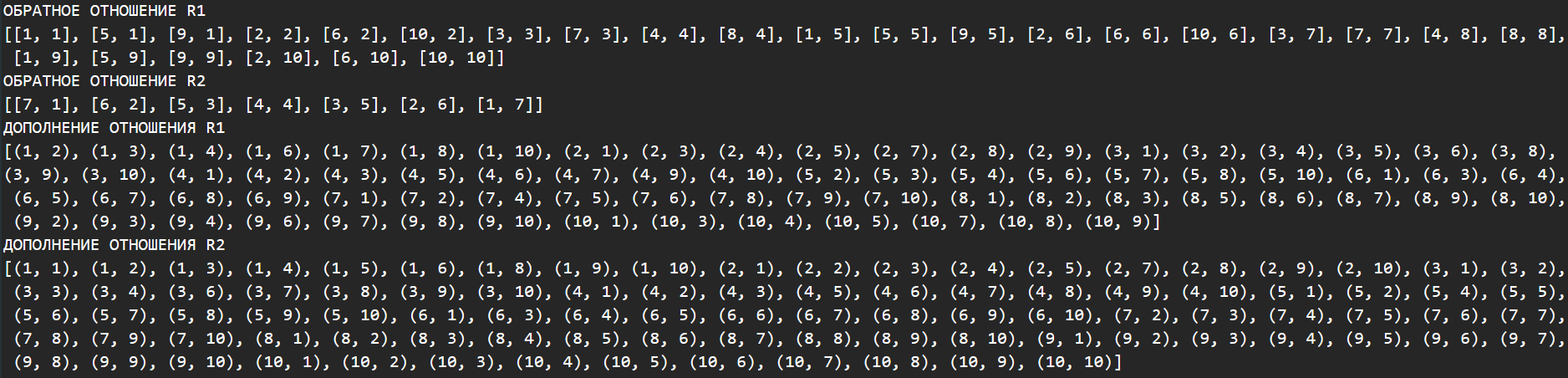
print("ДОПОЛНЕНИЕ ОТНОШЕНИЯ R1")

print(funcs.get\_relation\_addition(array = A, R = funcs.get\_R1(A)))

print("ДОПОЛНЕНИЕ ОТНОШЕНИЯ R2")

print(funcs.get\_relation\_addition(array = A, R = funcs.get\_R2(A)))

Результат выполнения:



3. Указать свойства отношений:

**Рефлексивность.** Отношение R на множестве V является рефлексивным, если для любого элемента v множества V, v ∈ V, следует, что (v, v) ∈ R, то есть пара (v, v) всегда принадлежит R. А отношение R на V не рефлексивно, если найдется такой элемент v ∈ V, что пара (v, v) ∉ R.

**Иррефлексивнось.** Отношение R на множестве V называется иррефлексивным (не путать с нерефлексивностью), если для каждого элемента v ∈ V следует, что (v, v) ∉ R.

**Симметричность.** Отношение R на множестве V называется симметричным, если вместе с (r1, r2) ∈ R всегда выполняется и (r2, r1) ∈ R.

**Асимметричность.** Отношение R на множестве V асимметрично, если для каждого набора (r1, r2) ∈ R, в котором r1 ≠ r2, справедливо, что (r2, r1) ∉ R.

**Транзитивность.** Отношение R на множестве V является транзитивным, если из включений (a, b) ∈ R и (b, c) ∈ R, всегда вытекает, что и (a, c) ∈ R.

**Отношение эквивалентности.** Отно­ше­нием эквивалентности является такое отношение, которое одновременно обладает свойствами рефлексивности, симметричности и транзитивности.

Свойства отношения R1: Рефлексивность, Симметричность, Транзитивность, Отношение эквивалентности.

Свойства отношения R2: Симметричность.

4. Для отношения, не обладающего свойством транзитивности, построить транзитивное замыкание алгоритмом *Флойда-Уоршолла*

Код программы:

Модуль funcs:

def get\_transitive\_closure(matrix):

n = len(matrix)

transitive\_closure = matrix

for k in range(n):

for i in range(n):

for j in range(n):

if (matrix[i][k] and matrix[k][j] == True):

transitive\_closure[i][j] = 1

return transitive\_closure

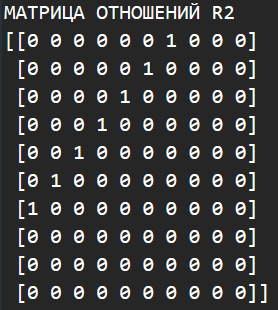
Модуль main:

print("ТРАНЗИТИВНОЕ ЗАМЫКАНИЕ ОТНОШЕНИЯ R2")

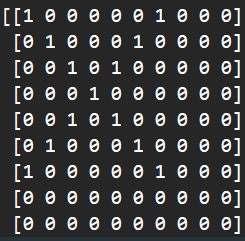
print(funcs.get\_transitive\_closure(matrix = funcs.matrix\_build(R = funcs.get\_R2(A), array = A))

Результат выполнения:

Исходная матрица отношений:



Транзитивное замыкание:



5. Найти композицию  или  отношений, указать обладает ли операция композиции отношений *свойством коммутативности*.

Код программы:

Модуль funcs:  
def attitudes\_composition(R1, R2):

composition = []

composition\_pair = []

for R1\_pair in R1:

for R2\_pair in R2:

if R1\_pair[1] == R2\_pair[0]:

composition\_pair.append(R1\_pair[0])

composition\_pair.append(R2\_pair[1])

composition.append(composition\_pair)

composition\_pair = []

new\_composition = []

for pair in composition:

if pair not in new\_composition:

new\_composition.append(pair)

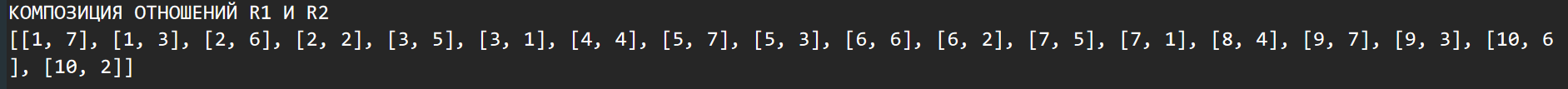
return new\_composition

Модуль main:

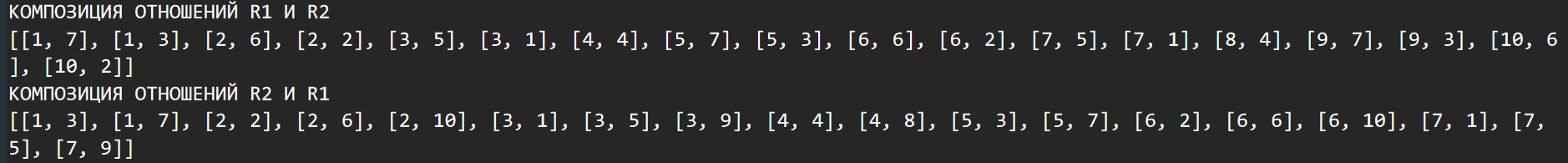
print("КОМПОЗИЦИЯ ОТНОШЕНИЙ R1 И R2")

print(funcs.attitudes\_composition(funcs.get\_R1(A), funcs.get\_R2(A)))

Результат выполнения:



Операция композиции отношений не обладает свойством коммутативности – R1 o R2 != R2 o R1

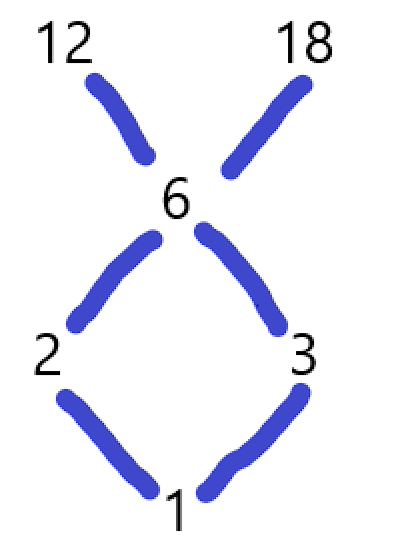
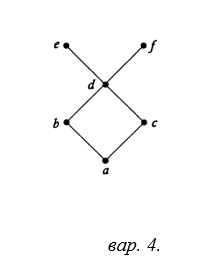


Задание 2:

На подмножестве *A* натуральных чисел *N*,  задано отношение частичного

порядка: *a* – делитель *b*. *ЧУ*-множество,  задано диаграммой *Хассе*.

1. Указать элементы ЧУМ 



{1, 2, 3, 6, 12, 18}

2. Пример сравнимых и несравнимых элементов

Несравнимы: 2, 3

Сравнимы: 1, 2; 1, 3; 2, 6; 3, 6; 6, 12; 6, 18;

3. Наибольший элемент: не существует

4. Наименьший элемент: 1

5. Максимальный элемент: 12, 18

Минимальный элемент: 1

6. Является ли рассматриваемое ЧУ-множество верхней или нижней полу

полурешеткой (или и тем, и другим)? – Нижняя полурешетка.

***Функции***

Вариант 10

Задание 1:

1. На множествах *B = {1,2,3,4}* и *C = {5,6,7,8}* задано отношение *S*  согласно вашему варианту. Определить (программная реализация) является ли заданное отношение *функцией* и если да, определить тип функции (*инъекция, сюръекция, биекция*). Результат проиллюстрировать графически

S = (1,8),(2,7),(3,5),(4,6)

Код программы:

#ФУНКЦИЯ ВОЗВРАЩАЕТ TRUE ЕСЛИ ОТНОШЕНИЕ ЯВЛЯЕТСЯ ФУНКЦИЕЙ

Модуль func:

def is\_func(S):

unambiguity = []

for pair in S:

if pair[0] in unambiguity:

return False

unambiguity.append(pair[0])

return True

Модуль main:

B = [1, 2, 3, 4]

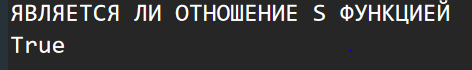
C = [5, 6, 7, 8]

S = [[1, 8], [2, 7], [3, 5], [4, 6]]

print("ЯВЛЯЕТСЯ ЛИ ОТНОШЕНИЕ S ФУНКЦИЕЙ")

print(funcs.is\_func(S))

Результат выполнения:



Задание 2.

*Задача.*

В коробке лежат *10* конфет в красных обертках, *10* в синих и *10* в желтых обертках. Какое наименьшее число конфет нужно вынуть не глядя, чтобы среди них обязательно оказались две конфеты в обертках,

*варианты*:

1. одного цвета;
2. разных цветов;
3. красного цвета;
4. не синего цвета?

(вариант рассчитывается по модулю *4*).

Вариант 2

Ответ: 10 + 1 = 11

Вывод: в ходе выполнения данной работы, я изучил базовые понятия теории множеств и операции над множествами.