МИНОБРНАУКИ РОССИИ
ФГБОУ ВО «СГУ ИМЕНИ Н. Г. ЧЕРНЫШЕВСКОГО»

ОТНОШЕНИЕ ЭКВИВАЛЕНТНОСТИ И ОТНОШЕНИЕ ПОРЯДКА

ЛАБОРАТОРНАЯ РАБОТА

студента 3 курса 331 группы
специальности 100501 — Компьютерная безопасность
факультета КНиИТ
Окунькова Сергея Викторовича

Проверил
аспирант _____ В. Н. Кутин

Саратов 2022

СОДЕРЖАНИЕ

1 Постановка задачи

Цель работы:

Изучение основных понятий универсальной алгебры и операций над бинарными отношениями.

Порядок выполнения работы:

1. Рассмотреть понятие алгебраической операции и классификацию свойств операций. Разработать алгоритмы проверки свойств операций: ассоциативность, коммутативность, идемпотентность, обратимость, дистрибутивность.
2. Рассмотреть основные операции над бинарными отношениями. Разработать алгоритмы выполнения операции над бинарными отношениями.
3. Рассмотреть основные операции над матрицами. Разработать алгоритмы выполнения операций над матрицами.

2 Теоретические сведения по рассмотренным темам с их обоснованием

2.1 Алгебраические операции

Опр. Отображение $f : A^n \to A$ называется алгебраической n-арной операцией или просто алгебраической операцией на множестве A. При этом n называется порядком или арностью алгебраической операции f.

Далее для бинарной операции f по возможности будем использовать мультипликативную запись с помощью символа $\cdot$, т.е.вместо f(x,y) писать $x \cdot y$.

Опр. Бинарная операция $\cdot$ на множестве A называется:

1. ассоциативной, если для любых $x, y, z \in A$ выполняется равенство
$$x \cdot (y \cdot z) = (x \cdot y) \cdot z;$$
2. коммутативной, если для любых $x, y \in A$ выполняется равенство
$$x \cdot y = y \cdot x;$$
3. идемпотентной, если для любого $x \in A$ выполняется равенство
$$x \cdot x = x;$$
4. обратимой, если для любого $x \in A$ и некоторого $a \in A$, выполняется свойство $x \cdot a = a \cdot x = 1$;
5. дистрибутивной относительно операции +, если для любых $x, y, z \in A$ выполняются равенства
$$x \cdot (y + z) = (x \cdot y) + (x \cdot z),$$
$$(y + z) \cdot x = (y \cdot x) + (z \cdot x);$$

2.2 Основные операции над бинарными отношениями

1. Теоретико-множественные операции ($\cup, \cap, \neg$)
2. Обращение бинарных отношений: обратным для бинарного отношения $\rho \subset A \times B$ называется бинарное отношение $\rho^{-1} \subset B \times A$, определяющееся по формуле:
$$\rho^{-1} = (b, a) : (a, b) \in \rho.$$
3. Композиция бинарных отношений: композицией бинарных отношений $\rho \subset A \times B$ и $\sigma \subset B \times C$ называется бинарное отношение $\rho\sigma \subset A \times C$, определяющееся по формуле:
$$\rho\sigma = (a, c) : (a, b) \in \rho \text{ и } (b, c) \in \sigma \text{ для некоторого } b \in B.$$

2.3 Основные операции над матрицами

1. Сложение и вычитание матриц.

   Суммой $A + B$ матриц $A_{m\times n} = (a_{ij})$ и $B_{m\times n} = (b_{ij})$ называется матрица $C_{m\times n} = (c_{ij})$, где $c_{ij} = a_{ij} + b_{ij}$ для всех $i = \overline{1, m}$ и $j = \overline{1, n}$.

   Разностью $A - B$ матриц $A_{m\times n} = (a_{ij})$ и $B_{m\times n} = (b_{ij})$ называется матрица $C_{m\times n} = (c_{ij})$, где $c_{ij} = a_{ij} - b_{ij}$ для всех $i = \overline{1, m}$ и $j = \overline{1, n}$.

2. Умножение матрицы на число.

   Произведением матрицы $A_{m\times n} = (a_{ij})$ на число α называется матрица $C_{m\times n} = (c_{ij})$, где $c_{ij} = \alpha a_{ij}$ для всех $i = \overline{1, m}$ и $j = \overline{1, n}$.

3. Произведение двух матриц.

   Произведением матриц $A_{m\times n} = (a_{ij})$ на матрицу $B_{m\times n} = (b_{ij})$ называется матрица $C_{m\times n} = (c_{ij})$, где $c_{ij} = \sum\limits_{p=1}^{n} a_{ip}b_{pj}$ для всех $i = \overline{1, m}$ и $j = \overline{1, n}$.

4. Транспонирование матрицы.

   Транспонированной по отношению к матрице $A_{m\times n} = (a_{ij})$ называется матрица $A^T_{n\times m} = (a^T_{ij})$ для элементов которой $a^T_{ij} = a_{ji}$.

3 Результаты работы

3.1 Описание алгоритмов проверки свойств операций

1. Алгоритм 1 - Проверка операции на ассоциативность:
   Вход: таблица Кэли операции $A = (a_{ij})$, размерности $n \times n$ и множество элементов S, размерности n, на котором задана операция
   Выход: "Операция является ассоциативной"или "Операция не является ассоциативной"
   Шаг 1. Зафиксировать элемент $s[i] \in S$ (изначально $i = 0$).
   Шаг 2. Создать новое множество S', размерностью n, где $s'[j] = a[i][j],$ где $0 \leq j < n$.
   Шаг 3. Создать таблицу Кэли A', размерностью $n \times n$, где $a'[j][g] = a[j][$индекс $s'[g]$ в множестве $s],$ где $0 \leq j, g < n$.
   Шаг 4. Создать новое множество S'', размерностью n, где $s''[j] = a[j][i],$ где $0 \leq j < n$.
   Шаг 5. Создать таблицу Кэли A'', размерностью $n \times n$, где $a''[j][g] = a[$индекс $s''[j]$ в множестве $s][g],$ где $0 \leq j, g < n$.
   Шаг 6. Сравнить на равенство A' и A'' ($a'[j][g] = a''[j][g],$ где $0 \leq j, g < n$). Если они не равны, вывести "Операция не является ассоциативной"и завершить выполнение алгоритма, иначе повторять шаги 1-6 пока $i < n$.
   Шаг 7. Вывести "Операция является ассоциативной".
   Асимптотика $O(n^3)$.
2. Алгоритм 2 - Проверка операции на коммутативность:
   Вход: таблица Кэли операции $A = (a_{ij})$
   Выход: "Операция является коммутативной"или "Операция не является коммутативной"
   Шаг 1. Транспонируем А, чтобы получить $B = A^T$ ($0 \leq i, j < n, b[i][j] \in B : b[i][j] = a[j][i]$).
   Шаг 2. Если $A = B$ ($b[i][j] \in B : b[i][j] = a[i][j]$, где $0 \leq i, j < n$), то вывести "Операция является коммутативной иначе вывести "Операция не является коммутативной".
   Асимптотика $O(n^2)$.
3. Алгоритм 3 - Проверка операции на идемпотентность:
   Вход: таблица Кэли операции $A = (a_{ij})$, размерности $n \times n$ и множество элементов S, размерности n, на котором задана операция

Выход: "Операция является идемпотентной"или "Операция не является идемпотентной"

Шаг 1. Для $0 \leq i < n$ проверить, что $a[i][i] = s[i]$. Если это условие выполняется, то вывести "Операция является идемпотентной иначе "Операция не является идемпотентной".

Асимптотика $O(n)$.

4. Алгоритм 4 - Проверка операции на обратимость:

   Вход: таблица Кэли операции $A = (a_{ij})$, размерности $n \times n$ и множество элементов S, размерности n, на котором задана операция

   Выход: Список обратных элементов матрицы B или "Операция не является обратимой"

   Шаг 1. Создать пустой список B

   Шаг 2. Зафиксировать элемент $s[i] \in S$ (изначально $i = 0$).

   Шаг 3. Для $0 \leq j < n$ проверить, что $a[i][j] = a[j][i] = 1$. Если это условие выполняется, то добавить $s[i]$ в список B.

   Шаг 4. Повторять шаги 2 и 3 пока $i < n$.

   Шаг 5. Если список B не пустой, то вернуть его, иначе вывести "Операция не является обратимой".

   Асимптотика $O(n^2)$.

5. Алгоритм 5 - Проверка операции на дистрибутивность:

   Вход: таблица Кэли операции 1 $A = (a_{ij})$, размерности $n \times n$, таблица Кэли операции 2 $A' = (a'_{ij})$, относительно которой делается проверка на дистрибутивность, размерности $n \times n$, множество элементов S, размерности n, на котором заданы обе операции

   Выход: "Операция 1 является дистрибутивной относительно операции 2"или "Операция 1 не является дистрибутивной относительно операции 2"

   Шаг 1. Если для любых $0 \leq i, j, g < n$ выполняется $a[i]$[индекс $a'[j][g]$) в множестве $S] = a'$[индекс $a[i][j]$ в множестве $S]$[индекс $a[i][g]$ в множестве $S]$ и a[индекс $a'[j][g]$ в множестве $S][i] = a'$[индекс $a[j][i]$ в множестве $S]$[индекс $a[g][i]$ в множестве $S]$, то вывести "Операция 1 является дистрибутивной относительно операции 2 иначе вывести "Операция 1 не является дистрибутивной относительно операции 2".

   Асимптотика $O(n^3)$.

3.2 Описание алгоритмов выполнения операций над бинарными отношениями

1. Алгоритм 6 - Операция объединения бинарных отношений:
   Вход: бинарное отношение $A = (a_{ij})$, размерности $n \times n$ и бинарное отношение $B = (b_{ij})$, размерности $n \times n$
   Выход: бинарное отношение $C = (c_{ij})$, размерности $n \times n$, представляющее собой объединение бинарного отношения A и бинарного отношения B
   Шаг 1. Создать матрицу C, размерности $n \times n$, и заполнить ее нулями.
   Шаг 2. Если $a[i][j] = 1$ или $b[i][j] = 1$, то $c[i][j]$ присвоить еденицу ($0 \leq i, j < n$).
   Шаг 3. Вернуть матрицу C.
   Асимптотика $O(n^2)$.
2. Алгоритм 7 - Операция пересечения бинарных отношений:
   Вход: бинарное отношение $A = (a_{ij})$, размерности $n \times n$ и бинарное отношение $B = (b_{ij})$, размерности $n \times n$
   Выход: бинарное отношение $C = (c_{ij})$, размерности $n \times n$, представляющее собой пересечение бинарного отношения A и бинарного отношения B
   Шаг 1. Создать матрицу C, размерности $n \times n$, и заполнить ее нулями.
   Шаг 2. Если $a[i][j] = 1$ и $b[i][j] = 1$, то $c[i][j]$ присвоить еденицу ($0 \leq i, j < n$).
   Шаг 3. Вернуть матрицу C.
   Асимптотика $O(n^2)$.
3. Алгоритм 8 - Операция инверсии бинарного отношения:
   Вход: бинарное отношение $A = (a_{ij})$, размерности $n \times n$
   Выход: бинарное отношение $C = (c_{ij})$, размерности $n \times n$, представляющее собой инверсию бинарного отношения A
   Шаг 1. Создать матрицу C, размерности $n \times n$, и заполнить ее нулями.
   Шаг 2. Если $a[i][j] = 1$, то $c[i][j]$ присвоить ноль, иначе $c[i][j]$ присвоить еденицу ($0 \leq i, j < n$).
   Шаг 3. Вернуть матрицу C.
   Асимптотика $O(n^2)$.
4. Алгоритм 9 - Операция обращения бинарного отношения:

Вход: бинарное отношение $A = (a_{ij})$, размерности $n \times n$

Выход: бинарное отношение $C = (c_{ij})$, размерности $n \times n$, представляющее собой обращение бинарного отношения A

Шаг 1. Создать матрицу C, размерности $n \times n$, и заполнить ее нулями.

Шаг 2. Если $a[i][j] = 1$, то $c[j][i]$ присвоить еденицу ($0 \leq i, j < n$).

Шаг 3. Вернуть матрицу C.

Асимптотика $O(n^2)$.

5. Алгоритм 10 - Операция композиции бинарного отношения:

Вход: бинарное отношение $A = (a_{ij})$, размерности $n \times m$ и бинарное отношение $B = (b_{ij})$, размерности $m \times h$

Выход: бинарное отношение $C = (c_{ij})$, размерности $n \times h$, представляющее собой композицию бинарных отношений A и B

Шаг 1. Создать матрицу C, размерности $n \times h$, и заполнить ее нулями.

Шаг 2. Если $a[i][j] = b[j][g] = 1$, то $c[i][g]$ присвоить еденицу ($0 \leq i < n, 0 \leq j < m, 0 \leq i < h$).

Шаг 3. Вернуть матрицу C.

Асимптотика $O(nmh)$.

3.3 Описание алгоритмов выполнения операций над матрицами

1. Алгоритм 11 - Операция сложения матриц: *Вход*: матрица $A = (a_{ij})$, размерности $n \times m$ и матрица $B = (b_{ij})$, размерности $n \times m$

Выход: матрица $C = (c_{ij})$, размерности $n \times m$, представляющую собой сумму A и B

Шаг 1. Создать матрицу C, размерности $n \times m$, и заполнить ее нулями.

Шаг 2. $c[i][j]$ присвоить $a[i][j] + b[i][j]$ ($0 \leq i < n, 0 \leq j < m$).

Шаг 3. Вернуть матрицу C.

Асимптотика $O(nm)$.

2. Алгоритм 12 - Операция вычитание матриц: *Вход*: матрица $A = (a_{ij})$, размерности $n \times m$ и матрица $B = (b_{ij})$, размерности $n \times m$

Выход: матрица $C = (c_{ij})$, размерности $n \times m$, представляющую собой разность A и B

Шаг 1. Создать матрицу C, размерности $n \times m$, и заполнить ее нулями.

Шаг 2. $c[i][j]$ присвоить $a[i][j] - b[i][j]$ ($0 \leq i < n, 0 \leq j < m$).

Шаг 3. Вернуть матрицу C.

Асимптотика $O(nm)$.

3. Алгоритм 13 - Операция произведения матрицы на константу: *Вход*: матрица $A = (a_{ij})$, размерности $n \times m$ и константа α
   Выход: матрица $C = (c_{ij})$, размерности $n \times m$, представляющую собой произведение A на α
   Шаг 1. Создать матрицу C, размерности $n \times n$, и заполнить ее нулями.
   Шаг 2. $c[i][j]$ присвоить $\alpha \cdot a[i][j]$ $(0 \leq i < n, 0 \leq j < m)$.
   Шаг 3. Вернуть матрицу C.
   Асимптотика $O(nm)$.
4. Алгоритм 14 - Операция произведения матриц: *Вход*: матрица $A = (a_{ij})$, размерности $n \times m$ и матрица $B = (b_{ij})$, размерности $m \times h$
   Выход: матрица $C = (c_{ij})$, размерности $n \times h$, представляющую собой произведение A и B
   Шаг 1. Создать матрицу C, размерности $n \times h$, и заполнить ее нулями.
   Шаг 2. $c[i][j]$ присвоить $\sum_{p=0}^{n-1} a[i][p]b[p][j]$ $(0 \leq i < n, 0 \leq j < m)$.
   Шаг 3. Вернуть матрицу C.
   Асимптотика $O(nmh)$.
5. Алгоритм 15 - Операция транспонирования матрицы:
   Вход: матрица $A = (a_{ij})$, размерности $n \times m$
   Выход: матрица $C = (c_{ij})$, размерности $n \times m$, представляющее собой транспонированную матрицу A
   Шаг 1. Создать матрицу C, размерности $m \times n$, и заполнить ее нулями.
   Шаг 2. $c[i][j]$ присвоить $a[j][i]$ $(0 \leq i < n, 0 \leq j < m)$.
   Шаг 3. Вернуть матрицу C.
   Асимптотика $O(n^2)$.

3.4 Коды программ, реализующей рассмотренные алгоритмы

```
import numpy as np


def get_set(a):
    s = []
    for i in range(len(a)):
        for j in range(len(a[i])):
            if a[i][j] == 1:
                s.append((i + 1, j + 1))
    return s
```

```
def get_matrix(st, n):
    a = np.zeros((n, n), int)
    for s in st:
        a[s[0] - 1][s[1] - 1] = 1
    return a


def inversion(a):
    n = len(a)
    for i in range(n):
        for j in range(n):
            if a[i][j] == 1:
                a[i][j] = 0
            else: a[i][j] = 1
    print(a)
    print(*get_set(a))


def dis(a):
    n = len(a)
    print("Enter second matrix")
    b = [list(map(int, input().split())) for _ in range(n)]
    c = get_set(a)
    c.extend(get_set(b))
    c = list(set(c))
    print(get_matrix(c, n))
    print(c)


def con(a):
    n = len(a)
    print("Enter second matrix")
    b = [list(map(int, input().split())) for _ in range(n)]
    a = np.array(a, int)
    b = np.array(b, int)
    c = a * b
    print(c)
    print(get_set(c))
```

```
def reverse(a):
    a = np.array(a, int)
    b = a.T
    print(b)
    print(*get_set(b))

def composition(a):
    n = len(a)
    print("Enter second matrix")
    b = [list(map(int, input().split())) for _ in range(len(a[0]))]
    c = np.zeros((n, len(b[0])), int)
    for i in range(len(a)):
        for j in range(len(a[i])):
            if a[i][j] == 1:
                for g in range(len(b[j])):
                    if b[j][g] == 1:
                        c[i][g] = 1
    print(c)
    print(get_set(c))

def task2():
    print('Enter the number of elements in relation')
    n = int(input())
    print("Enter your matrix")
    a = [list(map(int, input().split())) for _ in range(n)]
    print('What are you want to do with this binary relation?\n 1.Disjunction with an
    d = int(input())
    if d == 1:
        dis(a)
    elif d == 2:
        con(a)
    elif d == 3:
        inversion(a)
    elif d == 4:
        reverse(a)
    elif d == 5:
        composition(a)

def task3():
    print('Enter len of your matrix')
```

```
    n = int(input())
    print("Enter your matrix")
    a = np.array([list(map(int, input().split())) for _ in range(n)])
    print('What are you want to do with this matrix?\n 1.Transpose matrix\n 2.Add mat
    d = int(input())
    if d == 1:
        print(a.T)
    elif d == 4:
        print('Enter your constant')
        h = float(input())
        print(h * a)
    else:
        print("Enter another matrix")
        if d == 5:
            n1 = len(a[0])
        else: n1 = n
        b = np.array([list(map(int, input().split())) for _ in range(n1)])
        if d == 2:
            print(a + b)
        elif d == 3:
            print(a - b)
        elif d == 5:
            print(np.dot(a, b))


def isAssociative(matrix, st):
    flag = True
    for i in range(len(st)):
        st1 = [matrix[i][j] for j in range(len(st))]
        matrix1 = np.zeros((len(st), len(st)), int)
        for j in range(len(st)):
            for g in range(len(st)):
                matrix1[j][g] = matrix[j][st.index(st1[g])]
        st2 = [matrix[j][i] for j in range(len(st))]
        matrix2 = np.zeros((len(st), len(st)), int)
        for j in range(len(st)):
            for g in range(len(st)):
                matrix2[j][g] = matrix[st.index(st2[j])][g]
        flag = (matrix1 == matrix2).all()
        if not(flag):
            break
```

```
    if flag:
        print('This operation is associative!')
    else:
        print('This operation is not associative!')


def isCommutative(matrix):
    matrix1 = matrix.T
    if (matrix1 == matrix).all():
        print('This operation is commutative!')
    else:
        print('This operation is not commutative!')


def isIdempotent(matrix, st):
    flag = True
    for i in range(len(matrix)):
        if not(matrix[i][i] == st[i]):
            flag = False
            break
    if flag:
        print('This operation is idempotent!')
    else:
        print('This operation is not idempotent!')


def isReversible(matrix, st):
    rev = []
    for i in range(len(st)):
        flag = True
        for j in range(len(st)):
            if not(matrix[i][j] == matrix[j][i] == 1):
                flag = False
                break
        if flag:
            rev.append(st[i])
    if rev:
        print(f'Reverse elements of the operation: {rev}')
    else:
        print('This operation is not reversible!')
```

```
def isDistributive(matrix, st):
    print('Enter Cayley table for another operation')
    print("  ", *st)
    matrix1 = []
    for i in range(len(st)):
        print(st[i], end="  ")
        s = list(map(int, input().split()))
        matrix1.append(s)
    matrix1 = np.array(matrix1)
    flag = True
    for i in range(len(st)):
        for j in range(len(st)):
            for g in range(len(st)):
                if not(matrix[i][st.index(matrix1[j][g])] ==
                matrix1[st.index(matrix[i][j])][st.index(matrix[i][g])]
                and matrix[st.index(matrix1[j][g])][i] ==
                matrix1[st.index(matrix[j][i])][st.index(matrix[g][i])]):
                    flag = False
                    break
    if flag:
        print('This operation is distributive with respect to the introduced operation
    else:
        print('This operation is not distributive with respect to the introduced opera


def task1():
    print("Enter your set")
    st = list(map(int, input().split()))
    print('Enter Cayley table')
    print("  ", *st)
    matrix = []
    for i in range(len(st)):
        print(st[i], end="  ")
        s = list(map(int, input().split()))
        matrix.append(s)
    matrix = np.array(matrix)
    isAssociative(matrix, st)
    isCommutative(matrix)
    isIdempotent(matrix, st)
    isReversible(matrix, st)
```

```
    isDistributive(matrix, st)


def task4(lambda_digit=2):
    st = [1, 2, 3, 4]
    matrix = np.array([[1, 2, 1, 2],
                       [1, 2, 1, 2],
                       [1, 2, 3, 4],
                       [1, 2, 3, 4]])
    print("Task 1:")
    isAssociative(matrix, st)
    matrix_a = np.array([[1, -2], [-3, lambda_digit]])
    koef = lambda_digit / 2
    print("Task 2:")
    print(np.matmul(matrix_a, matrix_a) + (10 - koef) * matrix_a + koef * np.ones((2,
    matrix_a = np.array([[-1, lambda_digit, 3], [lambda_digit / 3, 2, 8 - lambda_digi
    matrix_b = np.array([[-lambda_digit, 2], [1, 10 - lambda_digit / 2], [-3, lambda_
    print("Task 3:")
    print(np.dot(matrix_a, matrix_b))



if __name__ == "__main__":
    print("What are you want? (1 - Checking properties of a binary operation, 2 - Per
    f = int(input())
    if f == 1:
        task1()
    elif f == 2:
        task2()
    elif f == 3:
        task3()
    elif f == 4:
        task4()
    else:
        print("Something going wrong! Enter a number from 1 to 3")
```

3.5 Результаты тестирования программы

```
What are you want? (1 - Checking properties of a binary operation, 2 - Perform an operation on a binary relation, 3 - Perform an operation on a matrix)
1
Enter your set
0 1 2
Enter Cayley table
   0 1 2
0  0 0 0
1  0 1 2
2  0 2 1
This operation is associative!
This operation is commutative!
This operation is not idempotent!
This operation is not reversible!
Enter Cayley table for another operation
   0 1 2
0  0 1 2
1  1 2 0
2  2 0 1
This operation is distributive with respect to the introduced operation!
```

Рисунок 1 – Тест алгоритма проверки свойств операций

```
What are you want? (1 - Checking properties of a binary operation, 2 - Perform an operation on a binary relation, 3 - Perform an operation on a matrix)
2
Enter the number of elements in relation
4
Enter your matrix
1 0 0 1
0 1 1 0
1 0 0 0
0 0 0 1
What are you want to do with this binary relation?
 1.Disjunction with another binary relation
 2.Conjunction with another binary relation
 3.Inversion binary relation
 4.Reverse binary relation
 5.Composition with another binary relation
1
Enter second matrix
0 1 1 0
1 0 0 0
0 1 0 1
1 1 0 1
[[1 1 1 1]
 [1 1 1 0]
 [1 1 0 1]
 [1 1 0 1]]
[(4, 4), (1, 2), (2, 1), (3, 4), (4, 1), (3, 1), (1, 1), (1, 4), (4, 2), (2, 3), (2, 2), (3, 2), (1, 3)]
```

Рисунок 2 – Тест алгоритма объединения бинарных отношений

```
What are you want? (1 - Checking properties of a binary operation, 2 - Perform an operation on a binary relation, 3 - Perform an operation on a matrix)
2
Enter the number of elements in relation
5
Enter your matrix
1 0 0 1 0
0 1 1 0 1
1 0 0 0 1
0 1 1 0 1
1 0 0 0 0
What are you want to do with this binary relation?
 1.Disjunction with another binary relation
 2.Conjunction with another binary relation
 3.Inversion binary relation
 4.Reverse binary relation
 5.Composition with another binary relation
2
Enter second matrix
0 1 1 0 1
1 0 0 1 1
1 1 1 1 1
0 1 0 1 1
1 0 1 1 0
[[0 0 0 0 0]
 [0 0 0 0 1]
 [1 0 0 0 1]
 [0 1 0 0 1]
 [1 0 0 0 0]]
[(2, 5), (3, 1), (3, 5), (4, 2), (4, 5), (5, 1)]
```

Рисунок 3 – Тест алгоритма пересечения бинарных отношений

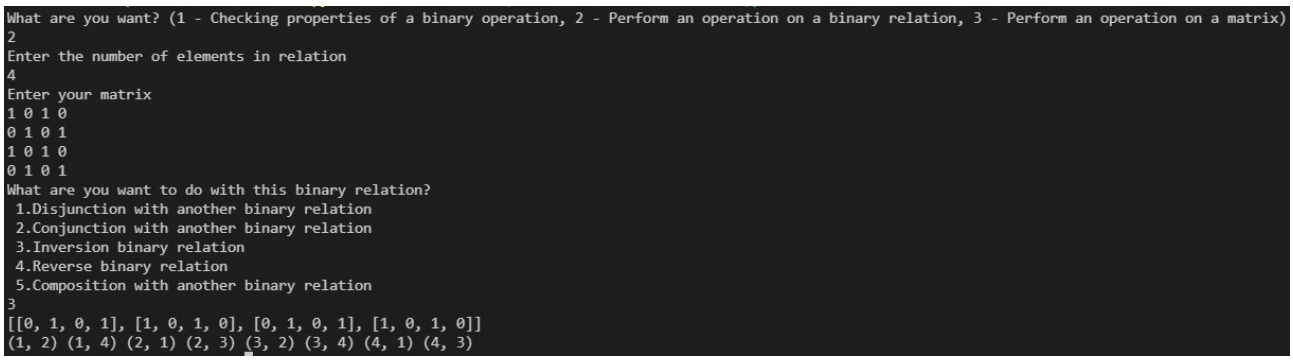

Рисунок 4 – Тест алгоритма инверсии бинарных отношений

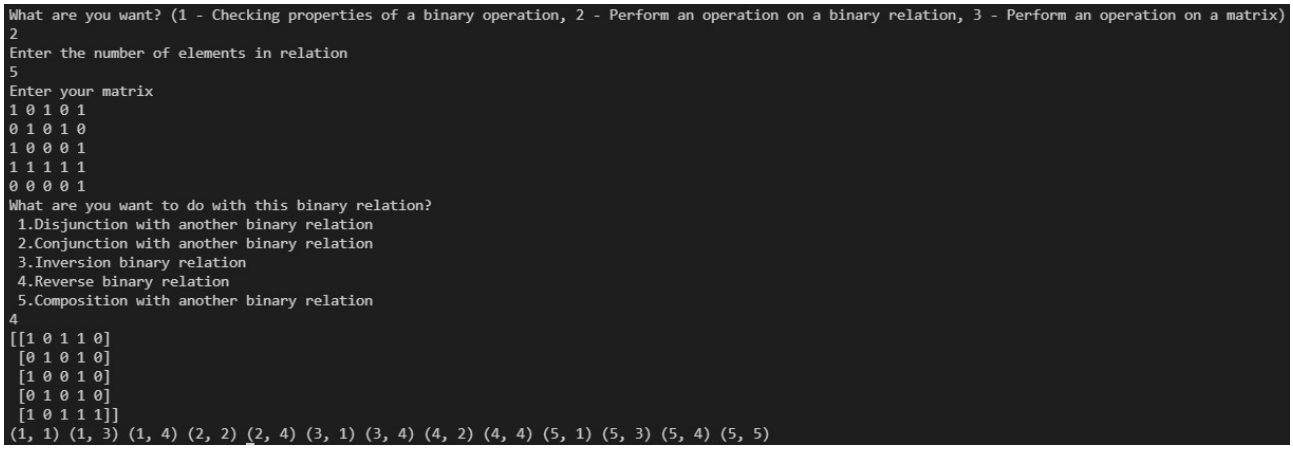

Рисунок 5 – Тест алгоритма обращения бинарных отношений

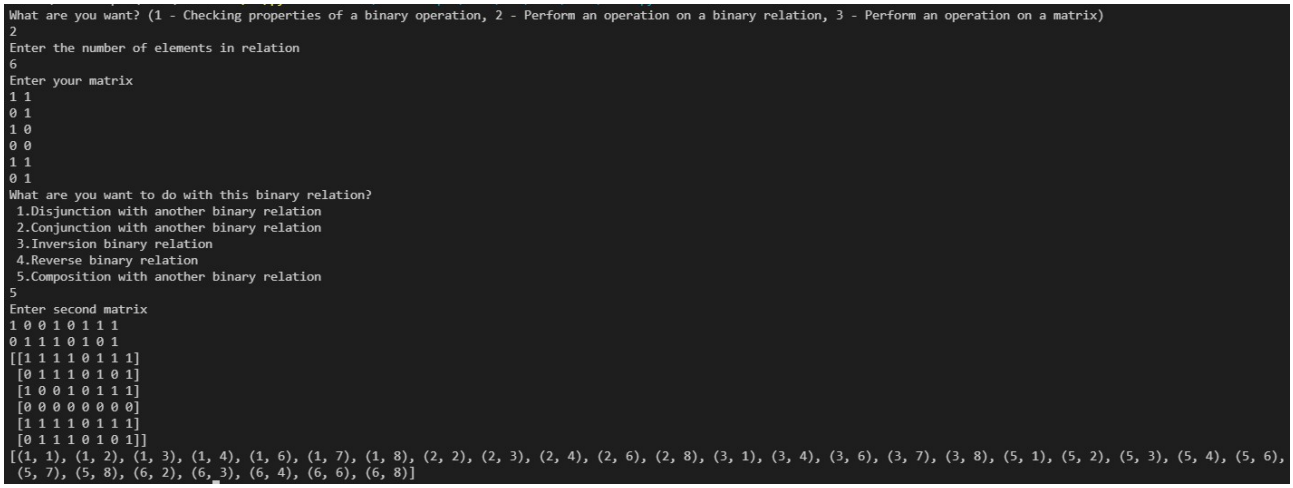

Рисунок 6 – Тест алгоритма композиции бинарных отношений

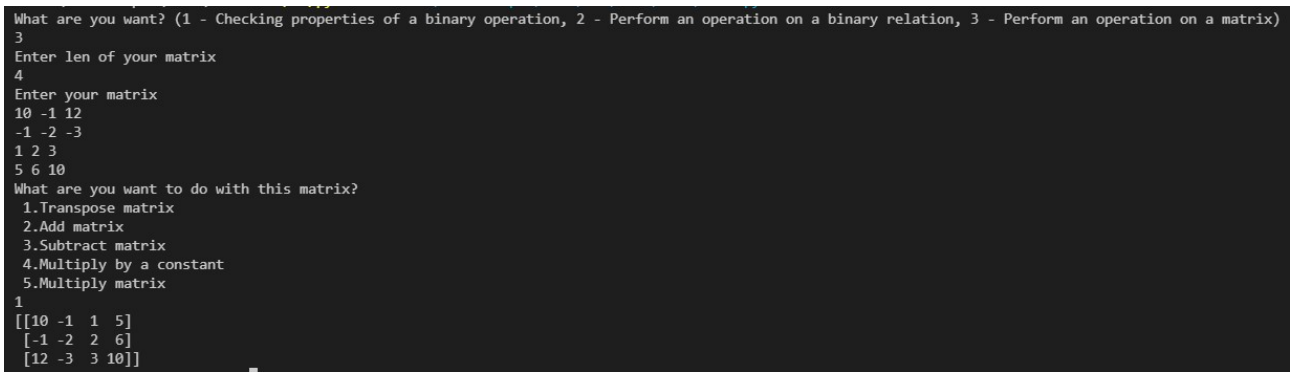

Рисунок 7 – Тест алгоритма транспонирования матриц

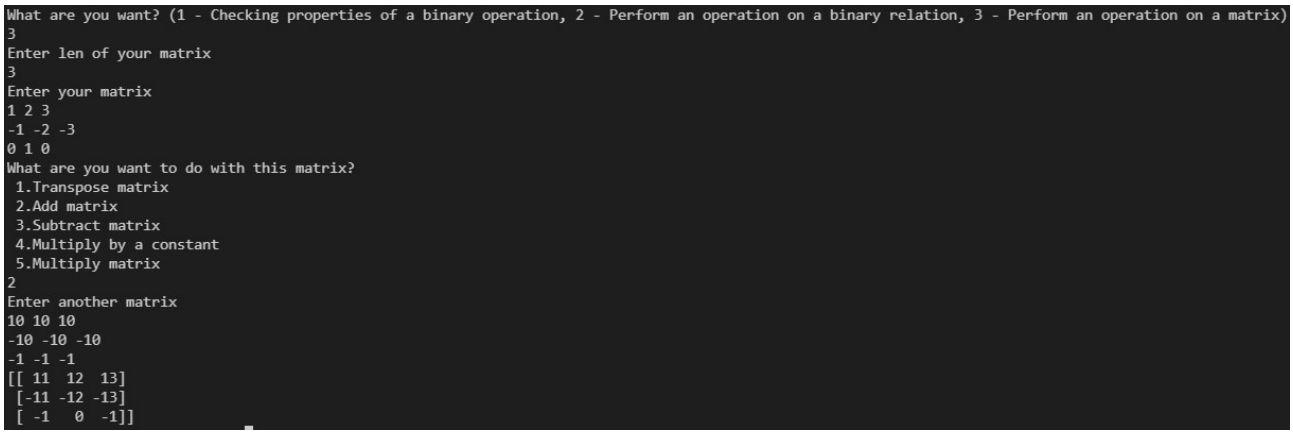

Рисунок 8 – Тест алгоритма сложения матриц

Рисунок 9 – Тест алгоритма вычитания матриц

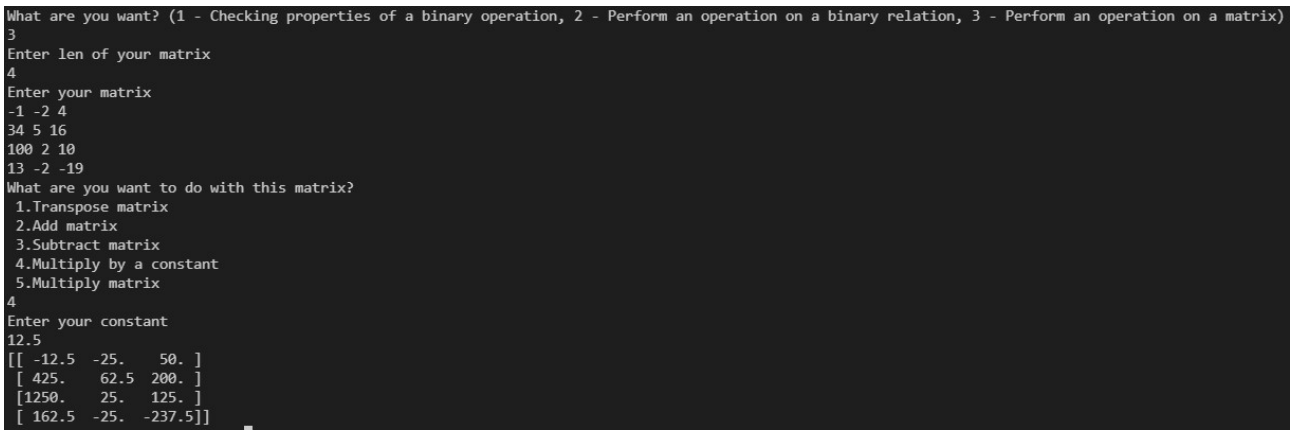

Рисунок 10 – Тест алгоритма умножение матрицы на костанту

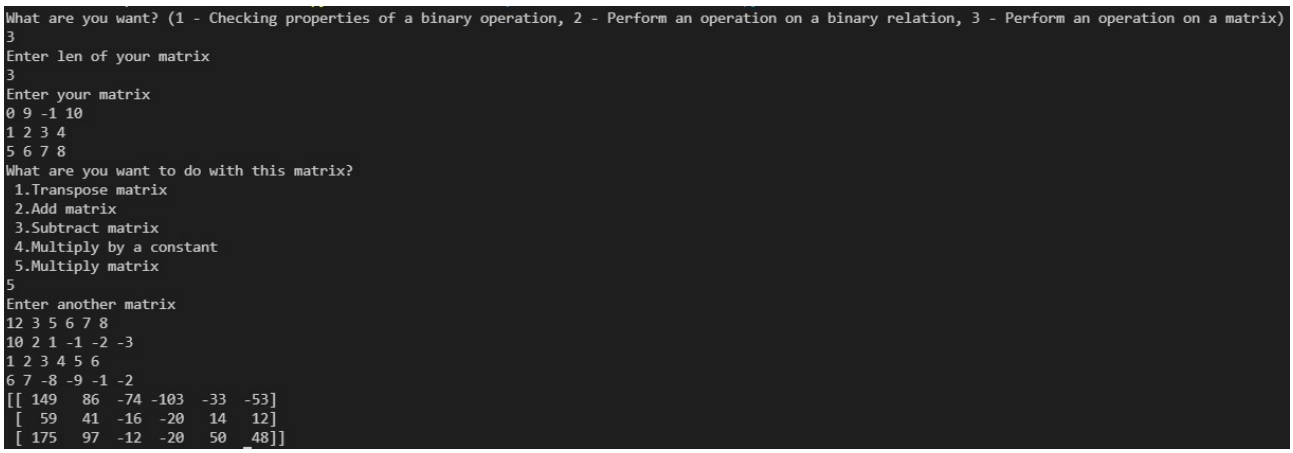

Рисунок 11 – Тест алгоритма умножение матриц

3.6 Оценки сложности рассмотренных алгоритмов

3.6.1 Проверка операции на ассоциативность

Содержит две пары вложенных циклов для генерации двух таблиц Кэли в цикле по n. Следовательно асимптотика данного алгоритма $O(2n^3) = O(n^3)$.

3.6.2 Проверка операции на коммутативность

С учетом того, что операция транспонирования имеет временную сложность $O(n^2)$ и временной сложности поэлементного сравнения($O(n^2)$), асимптотика алгоритма $O(n^2)$.

3.6.3 Проверка операции на идемпотентность

В реализации алгоритма был использован один цикл, следовательно его временная сложность определяется как $O(n)$.

3.6.4 Проверка операции на обратимость

В реализации алгоритма были использован два цикла, один из которых был вложен в другой, следовательно его временная сложность определяется как $O(n^2)$.

3.6.5 Проверка операции на дистрибутивность

В реализации алгоритма были три вложенных друг в друга цикла, следовательно его временная сложность определяется как $O(n^3)$.

3.6.6 Операции объединения бинарных отношений

В реализации алгоритма были использован два цикла, один из которых был вложен в другой, для прохода по всем элементам двух матриц, следовательно его временная сложность определяется как $O(n^2)$.

3.6.7 Операции пересечениияния бинарных отношений

В реализации алгоритма были использован два цикла, один из которых был вложен в другой, для прохода по всем элементам двух матриц, следовательно его временная сложность определяется как $O(n^2)$.

3.6.8 Операции инверсии бинарного отношений

В реализации алгоритма были использован два цикла, один из которых был вложен в другой, для прохода по всем элементам матрицы, следовательно его временная сложность определяется как $O(n^2)$.

3.6.9 Операции обращения бинарного отношений

В реализации алгоритма были использован два цикла, один из которых был вложен в другой, для прохода по всем элементам матрицы, следовательно его временная сложность определяется как $O(n^2)$.

3.6.10 Операции композиции бинарных отношений

В реализации алгоритма были три вложенных друг в друга цикла (один по n, вотро по m, третий по h), следовательно его временная сложность определяется как $O(nmh)$.

3.6.11 Операции сложения матриц

В реализации алгоритма были использован два цикла, один из которых был вложен в другой, для прохода по всем элементам двух матриц, следовательно его временная сложность определяется как $O(nm)$.

3.6.12 Операции вычитания матриц

В силу своей схожести с предыдущим алгоритмом временная сложность определяется как $O(nm)$.

3.6.13 Операции произведение матрицы на константу

В реализации алгоритма были использован два цикла, один из которых был вложен в другой, для прохода по всем элементам матрицы, следовательно его временная сложность определяется как $O(nm)$.

3.6.14 Операции транспонирования матрицы

В реализации алгоритма были использован два цикла, один из которых был вложен в другой, для прохода по всем элементам матрицы, следовательно его временная сложность определяется как $O(nm)$.

3.6.15 Операции произведения матриц

В реализации алгоритма были использован два цикла, один из которых был вложен в другой, для прохода по всем элементам первой матрицы, так же внутри вложенного цикла был вложен еще один цикл, с помощью которого считалась сума на i строке первой матрицы и j столбце второй, следовательно его временная сложность определяется как $O(nmh)$.

3.7 Ответы на задания:

```
What are you want? (1 - Checking properties of a binary operation, 2 - Perform an operation on a binary relation, 3 - Perform an operation on a matrix, 4 - Check tasks)
4
Task 1:
This operation is associative!
Task 2:
[[ 17. -23.]
 [-35.  29.]]
Task 3:
[[ -5.          22.        ]
 [-21.33333333  34.        ]]
```

Рисунок 12 – Ответы на задания, представленные в конце лабораторной

ЗАКЛЮЧЕНИЕ

В рамках данной лабораторной работы были рассмотренны понятие алгебраической операции и классификация ее свойств, основные операции над

бинарными отношениями, основные операции над матрицами. На основе этой теоретической части была смоделирована программа на языке Python с использованием средств библиотеки Numpy, которая способна проверки свойства заданной через таблицу Кэли операции, выполнить любую из основных операций над бинарым отношением или матрицей, а так же была оценена асимптотика каждого реализованного алгоритма.