

**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

## СОДЕРЖАНИЕ

1	Постановка задачи.....	3
2	Теоретические сведения по рассмотренным темам с их обоснованием ...	4
3	Результаты работы.....	6
3.1	Описание алгоритма классификации бинарных отношений.....	6
3.2	Описание алгоритмов построения основных замыканий бинарных отношений.....	8
3.3	Коды программ, реализующей рассмотренные алгоритмы .....	9
3.4	Результаты тестирования программ .....	16
3.5	Оценки сложности рассмотренных алгоритмов.....	18
3.5.1	Алгоритм определения рефлексивности .....	18
3.5.2	Алгоритм определения антирефлексивности .....	18
3.5.3	Алгоритм определения симметричности.....	18
3.5.4	Алгоритм определения антисимметричности .....	18
3.5.5	Алгоритм определения транзитивности .....	19
3.5.6	Алгоритм классификации .....	19
3.5.7	Построение замыкания рефлексивности .....	19
3.5.8	Построение замыкания симметричности .....	19
3.5.9	Построение замыкания транзитивности .....	19
	ЗАКЛЮЧЕНИЕ .....	20

## **1 Постановка задачи**

Цель работы:

Изучение основных свойств бинарных отношений и операций замыкания бинарных отношений.

Порядок выполнения работы:

1. Разобрать определения отношения эквивалентности, фактор-множества. Разработать алгоритмы построения эквивалентного замыкания бинарного отношения и системы представителей фактор-множества.
2. Разобрать определения отношения порядка и диаграммы Хассе. Разработать алгоритмы вычисления минимальных (максимальных) и наименьших (наибольших) элементов и построения диаграммы Хассе.
3. Разобрать определения контекста и концепта. Разработать алгоритм вычисления решетки концептов.

## 2 Теоретические сведения по рассмотренным темам с их обоснованием

Бинарное отношение  $\varepsilon$  на множестве  $A$  называется отношением эквивалентности (или просто эквивалентностью), если оно рефлексивно, симметрично и транзитивно.

Для любого подмножества  $X \subset A$  множество  $\rho(X) = \{b \in B : (x, b) \in \rho \text{ для некоторого } x \in X\}$  называется образом множества  $X$  относительно отношения  $\rho$ .

Образ одноэлементного множества  $X = \{a\}$  относительно отношения  $\rho$  обозначается символом  $\rho(a)$  и называется также образом элемента  $a$  или **срезом** отношения  $\rho$  через элемент  $a$ .

Срезы  $\varepsilon(a)$  называются классами эквивалентности по отношению  $\varepsilon$  и сокращенно обозначаются символом  $[a]$ . Множество всех таких классов эквивалентности  $\{[a] : a \in A\}$  называется фактор-множеством множества  $A$  по эквивалентности  $\varepsilon$  и обозначается символом  $A/\varepsilon$ .

Бинарное отношение  $\omega$  на множестве  $A$  называется отношением порядка (или просто порядком), если оно рефлексивно, антисимметрично и транзитивно.

Множество  $A$  с заданным на нем отношением порядка  $\leq$  называется упорядоченным множеством и обозначается  $A = (A, \leq)$  или просто  $(A, \leq)$ .

Элемент  $a$  упорядоченного множества  $(A, \leq)$  называется:

1. минимальным, если  $(\forall x \in A) x \leq a \implies x = a$ ,
2. максимальным, если  $(\forall x \in A) a \leq x \implies x = a$ ,
3. наименьшим, если  $(\forall x \in A) a \leq x$ ,
4. наибольшим, если  $(\forall x \in A) x \leq a$ .

Упорядоченное множество  $A = (A, \leq)$  наглядно представляется диаграммой Хассе, которая представляет элементы множества  $A$  точками плоскости и пары  $a < b$  представляет линиями, идущими вверх от элемента  $a$  к элементу  $b$ .

Алгоритм построения диаграммы Хассе конечного упорядоченного множества  $A = (A, \leq)$ .

1. В упорядоченном множестве  $A = (A, \leq)$  найти множество  $A_1$  всех минимальных элементов и расположить их в один горизонтальный ряд (это первый уровень диаграммы).
2. В упорядоченном множестве  $A \setminus A_1$ , найти множество  $A_2$  всех минимальных элементов и расположить их в один горизонтальный ряд над первым

уровнем (это второй уровень диаграммы). Соединить отрезками элементы этого ряда с покрываемыми ими элементами предыдущего ряда.

3. В упорядоченном множестве  $A \setminus (A_1 \cup A_2)$  найти множество  $A_3$  всех минимальных элементов и расположить их в один горизонтальный ряд над вторым уровнем (это третий уровень диаграммы). Соединить отрезками элементы этого ряда с покрываемыми ими элементами предыдущих рядов.
4. Процесс продолжается до тех пор, пока не будут выбраны все элементы множества  $A$ .

Контекстом называется алгебраическая система  $K = (G, M, \rho)$ , состоящая из множества объектов  $G$ , множества атрибутов  $M$  и бинарного отношения  $\rho \subset G \times M$ , показывающего  $(g, m) \in \rho$ , что объект  $g$  имеет атрибут  $m$ .

Упорядоченная пара  $(X, Y)$  замкнутых множеств  $X \in Z_{f_G}, Y \in Z_{f_M}$ , удовлетворяющих условиям  $\varphi(X) = Y, \psi(Y) = X$ , называется концептом контекста  $K = (G, M, \rho)$ . При этом компонента  $X$  называется объемом и компонента  $Y$  - содержанием концепта  $(X, Y)$ .

Множество всех концептов  $C(K)$  так упорядочивается отношением  $(X, Y) \leq (X_1, Y_1) \Leftrightarrow X \subset X_1$  (или равносильно  $Y_1 \subset Y$ ), что  $(C(K), \leq)$  является полной решеткой, которая изоморфна решетке замкнутых подмножеств множества  $G$ .

Алгоритм вычисления системы замыканий на множестве  $G$ :

1. Рассматриваем множество  $G \in Z_{f_G}$ .
2. Последовательно перебираем все элементы  $m \in M$  и вычисляем для них  $\psi(\{m\}) = \rho^{-1}(m)$ .
3. Вычисляем все новые пересечения множества  $\psi(\{m\})$  с ранее полученными множествами и добавляем новые множества к  $Z_{f_G}$ . Аналогично вычисляется система замыканий на множестве  $M$ .

### 3 Результаты работы

#### 3.1 Описание алгоритма классификации бинарных отношений

##### 1. Алгоритм 1 - Проверка бинарного отношения на рефлексивность:

Бинарное отношение называется рефлексивным тогда и только тогда, когда  $\Delta_A \subset \rho$ . Это означает, что бинарное отношение  $\rho$  рефлексивно, если  $M(\rho) \geq E$ , где  $E$  - единичная матрица. Если же матрица  $M(\rho)$  несравнима с единичной матрицей, то бинарное отношение  $\rho$  не является рефлексивным;

*Вход:* матрица бинарного отношения  $A = (a_{ij})$ , размерности  $n \times n$

*Выход:* "Множество рефлексивно" или "Множество не рефлексивно"

Шаг 1. Суммирование элементов на главной диагонали ( $sum = \sum_{i=1}^n a[i][i]$ ).

Шаг 2. Если  $sum = n$ , то отношение является рефлексивным, иначе не рефлексивным.

Асимптотика  $O(n)$ .

##### 2. Алгоритм 2 - Проверка бинарного отношения на симметричность:

Бинарное отношение называется симметричным тогда и только тогда, когда  $\rho^{-1} \subset \rho$ . Это означает, что бинарное отношение  $\rho$  симметрично, если  $M(\rho) \geq M(\rho)^T$ , где  $M(\rho)^T$  – транспонированная матрица бинарного отношения  $\rho$ . Если же матрица  $M(\rho)$  несравнима с  $M(\rho)^T$ , то бинарное отношение  $\rho$  не является симметричным;

*Вход:* матрица бинарного отношения  $A = (a_{ij})$ , размерности  $n \times n$

*Выход:* "Множество симметрично" или "Множество не симметрично"

Шаг 1. Транспонируем  $A$ , 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 бинарное отношение будет является симметричным, иначе отношение не симметрично.

Асимптотика  $O(n^{3/2} \log n)$ .

##### 3. Алгоритм 3 - Проверка бинарного отношения на транзитивность:

Бинарное отношение транзитивным тогда и только тогда, когда  $\rho\rho \subset \rho$ . Это означает, что бинарное отношение  $\rho$  транзитивно, если  $M(\rho)M(\rho) \leq M(\rho)$ .

*Вход:* матрица бинарного отношения  $A = (a_{ij})$ , размерности  $n \times n$

*Выход:* "Множество транзитивно" или "Множество не транзитивно"

На вход подается матрица бинарного отношения  $A$ .

Шаг 1. Получить матрицу  $B = A^2$ .

Шаг 2. Сравнить полученную и исходную матрицу.

Шаг 3. Если  $B \leq A$  ( $b[i][j] \in B : b[i][j] \leq a[i][j]$ , где  $0 \leq i, j < n$ ), то бинарное отношение транзитивно, иначе не транзитивным.

Асимптотика  $O(n^3)$ .

4. Алгоритм 4 - Проверка бинарного отношения на антирефлексивность:

*Вход:* матрица бинарного отношения  $A = (a_{ij})$ , размерности  $n \times n$

*Выход:* "Множество антирефлексивно" или "Множество не антирефлексивно"

На вход подается матрица бинарного отношения  $A$ .

Шаг 1. Суммирование элементов на главной диагонали ( $sum = \sum_{i=1}^n a[i][i]$ ).

Шаг 2. Если  $sum = 0$ , то отношение является антирефлексивным, иначе не антирефлексивным.

Асимптотика  $O(n)$ .

5. Алгоритм 5 - Проверка бинарного отношения на антисимметричность:

*Вход:* матрица бинарного отношения  $A = (a_{ij})$ , размерности  $n \times n$

*Выход:* "Множество антисимметрично" или "Множество не антисимметрично"

Шаг 1. Транспонируем  $A$ , чтобы получить  $C = A^T$ .

Шаг 2. Получим матрицу  $B = A * C$ .

Шаг 4. Если  $b[i][j] = 0$ , где  $b[i][j]$  элемент матрицы  $B$ ,  $0 \leq i, j < n$  и  $i \neq j$ , то отношение является антисимметричным, иначе отношение не антисимметрично.

Асимптотика  $O(n^3)$ .

6. Алгоритм 6 - Классификация бинарного отношения:

*Вход:* матрица бинарного отношения  $A = (a_{ij})$ , размерности  $n \times n$

*Выход:* «Бинарное отношение является отношением квазипорядка», «Бинарное отношение является отношением эквивалентности», «Бинарное отношение является отношением частичного порядка» или «Бинарное отношение является отношением строгого порядка».

Шаг 1. Запустить алгоритмы 1 и 3 (проверки на рефлексивность и транзитивность), подав им на вход матрицу  $A$ . Если алгоритмы вернут значения «Бинарное отношение является рефлексивным» и «Бинарное отношение

является транзитивным», то вернуть значение «Бинарное отношение является отношением квазипорядка».

Шаг 2. Запустить алгоритмы 1, 2 и 3 (проверки на рефлексивность, симметричность и транзитивность), подав им на вход матрицу  $A$ . Если алгоритмы вернут значения «Бинарное отношение является рефлексивным», «Бинарное отношение является симметричным» и «Бинарное отношение является транзитивным», то вернуть значение «Бинарное отношение является отношением эквивалентности».

Шаг 3. Запустить алгоритмы 3 и 5 (проверки на антисимметричность и транзитивность), подав им на вход матрицу  $A$ . Если алгоритмы вернут значения «Бинарное отношение является антисимметричным» и «Бинарное отношение является транзитивным», то вернуть значение «Бинарное отношение является отношением частичного порядка».

Шаг 4. Запустить алгоритмы 3, 4 и 5 (проверки на антирефлексивность, антисимметричность и транзитивность), подав им на вход матрицу  $A$ . Если алгоритмы вернут значения «Бинарное отношение является антирефлексивным», «Бинарное отношение является антисимметричным» и «Бинарное отношение является транзитивным», то вернуть значение «Бинарное отношение является отношением строгого порядка».

Если не учитывать сложность вызываемых алгоритмов, то асимптотика  $O(1)$ , иначе асимптотика  $O(n^3)$ .

### **3.2 Описание алгоритмов построения основных замыканий бинарных отношений**

1. Алгоритм 7 - Замыкание бинарного отношения относительно рефлексивности:

*Вход:* матрица бинарного отношения  $A = (a_{ij})$ , размерности  $n \times n$

*Выход:* матрица бинарного отношения  $A'$ , замкнутая относительно рефлексивности

Шаг 1. Присвоить каждому  $a[i][i]$  значение 1, где  $0 \leq i < n$ , после чего вернуть полученную матрицу бинарного отношения  $A' = (a'_{ij})$  с построенным на нем рефлексивным замыканием.

Асимптотика  $O(n)$ .

2. Алгоритм 8 - Замыкание бинарного отношения относительно симметричности:



*Вход:* матрица бинарного отношения  $A = (a_{ij})$ , размерности  $n \times n$

*Выход:* матрица бинарного отношения  $A'$ , замкнутая относительно симметричности

Шаг1. Каждому элементу  $a[i][j]$   $0 \leq i, j < n$  матрицы  $A$  присваивается значение элемент  $a[j][i]$ , после чего вернуть полученную матрицу бинарного отношения  $A' = (a'_{ij})$  с построенным на нем симметричным замыканием.

Асимптотика  $O(n^2)$ .

### 3. Алгоритм 9 - Замыкание бинарного отношения относительно транзитивности:

*Вход:* матрица бинарного отношения  $A = (a_{ij})$ , размерности  $n \times n$

*Выход:* матрица бинарного отношения  $A'$ , замкнутая относительно транзитивности

Шаг1. Если  $a[i][k] = 1$  и  $a[k][j] = 1$ , то присвоить  $a[i][j]$  значение 1, где  $0 \leq i, j, k < k$ . Такой шаг нужно повторить  $n$  раз в силу определения п.3) оператора транзитивного замыкания в лемме 2, после чего вернуть полученную матрицу бинарного отношения  $A' = (a'_{ij})$  с построенным на нем транзитивным замыканием.

Асимптотика  $O(n^4)$ .

## 3.3 Коды программ, реализующей рассмотренные алгоритмы

```
import numpy as np
import json

def isTran(a):
    n = len(a)
    b = np.matmul(a, a)
    for i in range(n):
        for j in range(n):
            if b[i][j]:
                b[i][j] = b[i][j] / b[i][j]
    f = (a >= b).all()
    if f:
        print("Set is transitive")
        return 't'
    else:
        f1 = True
```

```

    for i in range(n):
        for j in range(n):
            for k in range(n):
                if a[i][k] and a[k][j] and a[i][j]:
                    f1 = False

    if f1:
        print("Set is anti-transitive")
        return 'at'
    else:
        print("Set is not transitive")
        return 'nt'

def isSymm(a):
    b = a.transpose()
    if np.array_equal(a, b):
        print("Set is symmetry")
        return 's'
    else:
        f = True
        b = np.matmul(a, b)
        for i in range(len(b)):
            for j in range(len(b[i])):
                if b[i][j] != 0 and i != j:
                    f = False
                    break

        if f:
            print("Set is anti-symmetry")
            return 'as'
        else:
            print("Set is not symmetry")
            return 'ns'

def isRefl(a):
    n = len(a)
    sum = 0
    for i in range(n):
        sum += a[i][i]
    if sum == n:
        print("Set is reflexive")

```

```

        return 'r'
    elif sum == 0:
        print("Set is anti-reflexive")
        return 'ar'
    else:
        print("Set is not reflexive")
        return 'nr'

def makeRefl(a):
    n = len(a)
    for i in range(n):
        a[i][i] = 1

def makeSymm(a):
    n = len(a)
    for i in range(n):
        for j in range(n):
            if a[i][j]:
                a[j][i] = 1

def makeTran(a):
    n = len(a)
    for c in range(n):
        for k in range(n):
            for i in range(n):
                for j in range(n):
                    if a[i][k] and a[k][j]:
                        a[i][j] = 1

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_slices_list(relation_matrix, for_factor_set=1, attributes=None):

```

```

slices = []
for i in range(len(relation_matrix)):
    some_slice = [].copy()
    for j in range(len(relation_matrix[i])):
        if relation_matrix[i][j]:
            if attributes is None:
                some_slice.append(j + for_factor_set)
            else:
                some_slice.append(attributes[j])
    some_slice = tuple(some_slice)
    slices.append(some_slice)
return slices

def create_factor_set(bin_relation):
    slices = get_slices_list(bin_relation)
    print("Factor set:")
    factor_set = set(slices)
    print(factor_set)
    return factor_set

def create_representative_system(factor_set):
    representative_system = []
    factor_set = list(factor_set)
    factor_set.sort(key=len)

    elem_dict = {}

    for eq_class in factor_set:
        for representative in eq_class:
            if not(representative in representative_system):
                elem_dict[f"{representative}"] = eq_class
                representative_system.append(representative)
                break

    print("Representative set factor system:")
    print(representative_system)
    ans = ""
    print("Where:", ans)
    for k, v in elem_dict.items():

```

```

    print(f"{k} \u2208 {list(v)}; ")

def get_dividers(num):
    i = 1
    ans = []
    while(num >= i):
        if num % i == 0:
            ans.append(i)
        i += 1
    return ans

def task1():
    print("How you want enter your relation (set or matrix)?")
    enter = input()
    if enter == 'set':
        print('Enter the number of elements in relation')
        n = int(input())
        print("Enter your set")
        # st = [(1, 3), (3, 4), (1, 4), (2, 5), (5, 3)]
        s = list(map(str, input().split(' ')))
        st = []
        for c in s:
            a = ''
            i = 1
            while c[i] != ',':
                a += c[i]
                i += 1
            i += 1
            b = ''
            while c[i] != ')':
                b += c[i]
                i += 1
            st.append((int(a), int(b)))
        a = np.zeros((n, n), int)
        for s in st:
            a[s[0] - 1][s[1] - 1] = 1
        print('-----')
        print("Relation's matrix")
        print(a)
    else:

```

```

    print('Enter the number of elements in relation')
    n = int(input())
    print("Enter your matrix")
    a = [list(map(int, input().split())) for i in range(n)]
    a = np.array(a).reshape(n, n)
    print('-----')
    print("Relation's set")
    print(*get_set(a))
print('-----')
properties = {"reflexive": isRefl(a), "symmetry": isSymm(a), "transitive": isTran
print('-----')
if properties['reflexive'] == 'r' and properties['symmetry'] == 's' and properties
    print('Relation is equivalent')
    print('-----')
if properties['reflexive'] == 'r' and properties['transitive'] == 't':
    print('Relation is the relation of the quasi-order')
    print('-----')
if properties['reflexive'] == 'r' and properties['symmetry'] == 'as' and properti
    print('Relation is the relation of the partial order')
    print('-----')
if properties['reflexive'] == 'ar' and properties['symmetry'] == 'as' and properti
    print('Relation is the relation of the strict order')
    print('-----')
if properties['reflexive'] != 'r':
    makeRefl(a)
if properties['symmetry'] != 's':
    makeSymm(a)
if properties['transitive'] != 't':
    makeTran(a)
print('Closure matrix: ')
print(a)
print('-----')
print('Closure set: ')
print(*get_set(a))
print('-----')

factor_set = create_factor_set(a)
create_representative_system(factor_set)

def task2():
    print('Enter your number')

```

```
num = int(input())
dividers = get_dividers(num)
print(f'Dividers of {num} : {dividers}')

if __name__ == "__main__":
    task2()
```

### 3.4 Результаты тестирования программ

```
How you want enter your relation (set or matrix)?
set
Enter the number of elements in relation
3
Enter your set
(1,2) (1,3)
-----
Relation's matrix
[[0 1 1]
 [0 0 0]
 [0 0 0]]
-----
Set is anti-reflexive
Set is anti-symmetry
Set is transitive
-----
Relation is the relation of the strict order
-----
Closure matrix:
[[1 1 1]
 [1 1 0]
 [1 0 1]]
-----
Closure set:
(1, 1) (1, 2) (1, 3) (2, 1) (2, 2) (3, 1) (3, 3)
-----
```

Рисунок 1 – Ввод бинарного отношения (1, 2), (1, 3) в виде множества с выводом свойств этого множества и замыкания относительно эквивалентности



```

How you want enter your relation (set or matrix)?
set
Enter the number of elements in relation
5
Enter your set
(3,2) (2,5) (4,2)
-----
Relation's matrix
[[0 0 0 0 0]
 [0 0 0 0 1]
 [0 1 0 0 0]
 [0 1 0 0 0]
 [0 0 0 0 0]]
-----
Set is anti-reflexive
Set is not symmetry
Set is anti-transitive
-----
Closure matrix:
[[1 0 0 0 0]
 [0 1 1 1 1]
 [0 1 1 1 1]
 [0 1 1 1 1]
 [0 1 1 1 1]]
-----
Closure set:
(1, 1) (2, 2) (2, 3) (2, 4) (2, 5) (3, 2) (3, 3) (3, 4) (3, 5) (4, 2) (4, 3) (4, 4) (4, 5) (5, 2) (5, 3) (5, 4) (5, 5)
-----

```

Рисунок 2 – Ввод бинарного отношения (3, 2), (2, 5), (4, 2) в виде множества с выводом свойств этого множества и замыкания относительно эквивалентности

```

How you want enter your relation (set or matrix)?
matrix
Enter the number of elements in relation
5
Enter your matrix
0 0 0 0 0
0 0 0 0 1
0 1 0 0 0
0 1 0 0 0
0 0 0 0 0
-----
Relation's set
(2, 5) (3, 2) (4, 2)
-----
Set is anti-reflexive
Set is not symmetry
Set is anti-transitive
-----
Closure matrix:
[[1 0 0 0 0]
 [0 1 1 1 1]
 [0 1 1 1 1]
 [0 1 1 1 1]
 [0 1 1 1 1]]
-----
Closure set:
(1, 1) (2, 2) (2, 3) (2, 4) (2, 5) (3, 2) (3, 3) (3, 4) (3, 5) (4, 2) (4, 3) (4, 4) (4, 5) (5, 2) (5, 3) (5, 4) (5, 5)
-----

```

Рисунок 3 – Ввод бинарного отношения (3, 2), (2, 5), (4, 2) в виде матрицы с выводом свойств этого множества и замыкания относительно эквивалентности

```

How you want enter your relation (set or matrix)?
matrix
Enter the number of elements in relation
5
Enter your matrix
0 1 0 0 0
0 0 1 0 0
0 0 0 0 0
0 0 1 0 0
0 0 0 0 0
-----
Relation's set
(1, 2) (2, 3) (4, 3)
-----
Set is anti-reflexive
Set is not symmetry
Set is anti-transitive
-----
Closure matrix:
[[1 1 1 1 0]
 [1 1 1 1 0]
 [1 1 1 1 0]
 [1 1 1 1 0]
 [0 0 0 0 1]]
-----
Closure set:
(1, 1) (1, 2) (1, 3) (1, 4) (2, 1) (2, 2) (2, 3) (2, 4) (3, 1) (3, 2) (3, 3) (3, 4) (4, 1) (4, 2) (4, 3) (4, 4) (5, 5)
-----

```

Рисунок 4 – Ввод бинарного отношения (1, 2), (2, 3), (4, 3) в виде матрицы с выводом свойств этого множества и замыкания относительно эквивалентности

### 3.5 Оценки сложности рассмотренных алгоритмов

#### 3.5.1 Алгоритм определения рефлексивности

Сложность выполнения проверки на рефлексивность определяется как  $O(n)$ .

#### 3.5.2 Алгоритм определения антирефлексивности

За счет схожести с алгоритмом определения рефлексивности сложность определяется как  $O(n)$ .

#### 3.5.3 Алгоритм определения симметричности

Сложность транспонирования в numpy определяется как  $O(n^{3/2} \log n)$ , сложность сравнение двух матриц поэлементно определяется как  $O(n^2)$ . Отсюда можно сделать вывод, что сложность алгоритма будет определяться как  $O(n^{3/2} \log n + n^2) = O(n^{3/2} \log n)$ .

#### 3.5.4 Алгоритм определения антисимметричности

Сложность умножения матриц определяется как  $O(n^3)$ , сложность сравнение двух матриц поэлементно определяется как  $O(n^2)$ . Отсюда можно сделать

вывод, что сложность будет определяться как  $O(n^2 + n^3) = O(n^3)$

### 3.5.5 Алгоритм определения транзитивности

Из всего выше сказанного очевидно, что сложность проверки на транзитивность или антитранзитивность составляет  $O(n^3)$ , так как в нем используется умножение, сравнение матриц и тройной цикл для проверки рефлексивности в худшем случае матриц.

### 3.5.6 Алгоритм классификации

Сложность выполнения самого алгоритма классификации бинарных отношений реализованно через словарь языка python и оператор if, поэтому является константной ( $O(1)$ ), если не учитывать сложность выполнения проверки свойств отношения. Если учитывать сложность алгоритмов проверки свойств отношения, то :

1. Сложность проверка на квазипорядок определяется как  $O(n^3 + n) = O(n^3)$ .
2. Сложность проверка на эквивалентность определяется как  $O(n^3 + n + n^{3/2} \log n) = O(n^3)$ .
3. Сложность проверка на частичный порядок определяется как  $O(n^3 + n + n^3) = O(n^3)$ .
4. Сложность проверка на строгий порядок определяется как  $O(n^3 + n + n^3) = O(n^3)$ .

### 3.5.7 Построение замыкания рефлексивности

Так как весь алгоритм строится на заполнении главной диагонали матрицы 1, то его сложность составляет  $O(n)$ .

### 3.5.8 Построение замыкания симметричности

Для построения замыкания симметричности используются вложенный цикл, поэтому сложность алгоритма определяется как  $O(n^2)$ .

### 3.5.9 Построение замыкания транзитивности

Для построения замыкания транзитивности используются три вложенный цикла, поэтому сложность алгоритма определяется как  $O(n^4)$ .

## **ЗАКЛЮЧЕНИЕ**

В рамках данной лабораторной работы были рассмотрены теоритические основы свойств бинарных отношений, их видов и методов их замыкания по каждому из свойств. На основе этой теоретической части была смоделирована программа на языке Python с использованием средств библиотеки Numpy, которая способна определить свойства заданного множества, его вид и построить систему замыкания по каждому из основных свойств бинарного отношения, а так же была оценена асимптотика каждого реализованного алгоритма.