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

КЛАССИФИКАЦИЯ БИНАРНЫХ ОТНОШЕНИЙ И СИСТЕМЫ ЗАМЫКАНИЙ

ЛАБОРАТОРНАЯ РАБОТА

студента 3 курса 331 группы
направления 10.05.01 — Компьютерная безопасность
факультета КНиИТ
Никитина Арсения Владимировича

Проверил
ассистент ______________ Р. А. Фарахутдинов

Саратов 2022

СОДЕРЖАНИЕ

ВВЕДЕНИЕ . . . 3

1 **Цель работы и порядок ее выполнения** . . . 4

2 Теоретические сведения . . . 5

2.1 Основные определения видов бинарных отношений и их алгоритмы 5

2.1.1 Определение бинарного отношения . . . 5

2.1.2 Основные свойства бинарных и алгоритмы их определения . 5

2.2 Классификация бинарных отношений . . . 7

2.2.1 Определения классов бинарных отношений . . . 7

2.2.2 Алгоритм проверки отношения на квазипорядок . . . 7

2.2.3 Алгоритм проверки отношения на эквивалентность . . . 8

2.2.4 Алгоритм проверки отношения на частичный порядок . . . 8

2.3 Замыкания бинарных отношений и алгоритмы их построения . . . 8

2.3.1 Определение замыканий отношения . . . 8

2.3.2 Пример построения замыканий бинарного отношения . . . 9

2.3.3 Построение замыканий отношения . . . 9

3 Программная реализация рассмотренных алгоритмов . . . 12

3.1 Результаты тестирования программы . . . 12

3.2 Код программы, реализующей рассмотренные алгоритмы . . . 12

ЗАКЛЮЧЕНИЕ . . . 18

ВВЕДЕНИЕ

Существует определенная классификация бинарных отношений в зависимости от их свойств. Задачей данной работы является рассмотрение основных свойств бинарных отношений, а также их классификация. В зависимости от класса бинарного отношения, на нем можно определить замыкание: относительно рефлексивности, симметричности и транзитивности. Для этого требуется понимать, каким образом происходит классификация отношений в зависимости от множеств, которыми они могут задаваться.

1 Цель работы и порядок ее выполнения

Цель работы — изучение основных свойств бинарных отношений и операций замыкания бинарных отношений.

Порядок выполнения работы:

1. Разобрать основные определения видов бинарных отношений и разработать алгоритм классификации бинарных отношений.
2. Изучить свойства бинарных отношений и рассмотреть основные системы замыкания на множестве бинарных отношений.
3. Разработать алгоритмы построения основных замыканий бинарных отношений.

2 Теоретические сведения

2.1 Основные определения видов бинарных отношений и их алгоритмы

2.1.1 Определение бинарного отношения

Подмножества декартова произведения $A \times B$ множеств A и B называются **бинарными отношениями** между элементами множеств A и B и обозначаются строчными греческими буквами ρ, ρ_1 и т.п.

Для бинарного отношения $\rho \subset A \times B$ область определения D_ρ и множество значений E_ρ определяется как подмножества соответствущих множеств A и B по следующим формулам:

$$D_\rho = \{a : (a, b) \in \rho \text{ для некоторого } b \in B\},$$
$$E_\rho = \{b : (a, b) \in \rho \text{ для некоторого } a \in A\}.$$

2.1.2 Основные свойства бинарных и алгоритмы их определения

Бинарное отношение $\rho \subset A \times A$ называется:

1. *рефлексивным*, если $(a, a) \in \rho \; \forall a \in A$.
2. *антирефлексивным*, если $(a, a) \notin \rho \; \forall a \in A$.
3. *симметричным*, если $(a, b) \in \rho \Rightarrow (b, a) \in \rho \; \forall a, b \in A$.
4. *антисимметричным*, если $(a, b) \in \rho$ и $(b, a) \in \rho \; \Rightarrow a = b \; \forall a, b \in A$.
5. *транзитивным*, если $(a, b) \in \rho$ и $(b, c) \in \rho \Rightarrow (a, c) \in \rho \; \forall a, b, c \in A$.

Далее представлена программная реализация определения свойств бинарных отношений.

Пусть ρ - бинарное отношение на множестве $A = \{a_1, ..., a_N\}$ мощности N. Тогда *матрицей* бинарного отношения ρ будет матрица $M(\rho)$ размерности $N \times N$, определяемая следующим образом: $M(\rho)_{ij} = 1,$ если $(a_i, a_j) \in \rho,$ иначе $M(\rho)_{ij} = 0$

Выполним проверку свойств **рефлексивности** и **антирефлексивности**:

Алгоритм 1. Проверка бинарного отношения на *рефлексивность.*

Вход. Матрица $M(\rho)_{ij}$ бинарного отношения ρ размерности $N \times N$.

Выход. «Отношение является рефлексивным» или "Отношение не является рефлексивным".

1. Цикл по i от 1 до N.
2. Если $M_{ii} = 0$, то ответ «Отношение не является рефлексивным».
3. Если цикл завершен то ответ «Отношение является рефлексивным».

Трудоемкость алгоритма $O(n)$.

Алгоритм 2. Проверка бинарного отношения на *антирефлексивность.*

Вход. Матрица $M(\rho)_{ij}$ бинарного отношения ρ размерности $N \times N$.

Выход. «Отношение является антирефлексивным» или «Отношение не является антирефлексивным».

1. Цикл по i от 1 до N.
2. Если $M_{ii} = 1$, то ответ "Отношение не является антирефлексивным.
3. Если цикл завершен то ответ "Отношение является антирефлексивным.

Трудоемкость алгоритма $O(n)$.

Выполним проверку свойств **симметричности и антисимметричности**:

Свойство симметричности выполняется для отношения, заданного матрицей, если элементы, симметричные относительно главной диагонали равны.

Алгоритм 3. Проверка бинарного отношения на *симметричность.*

Вход. Матрица $M(\rho)_{ij}$ бинарного отношения ρ размерности $N \times N$.

Выход. «Отношение является симметричным» или «Отношение не является симметричным».

1. Цикл по i от 1 до N, цикл по j от 1 до N.
2. Если $M_{ij} \neq M_{ji}$, то ответ «Отношение не является симметричным».
3. Если циклы завершены, то ответ «Отношение является симметричным».

Трудоемкость алгоритма $O(N^2)$.

Алгоритм 4. Проверка бинарного отношения на *антисимметричность.*

Вход. Матрица $M(\rho)_{ij}$ бинарного отношения ρ размерности $N \times N$.

Выход. «Отношение является антисимметричным» или «Отношение не является антисимметричным».

1. Цикл по i от 1 до N, цикл по j от 1 до N.
2. Если $M_{ij} = M_{ji}$, то ответ «Отношение не является антисимметричным».
3. Если циклы завершены, то ответ «Отношение является антисимметричным».

Трудоемкость алгоритма $O(N^2)$.

Выполним проверку свойства **транзитивности**:

Свойство транзитивности выполняется для отношения, заданного матрицей, если для любого фиксированного элемента $M_{k,i} = 1$ из матрицы отношения, и для любого элемента из матрицы отношения $M_{i,j} = 1$, то выполняется $M_{k,j} = 1$.

Алгоритм 5. Проверка бинарного отношения на *транзитивность*.

Вход. Матрица $M(\rho)_{ij}$ бинарного отношения ρ размерности $N \times N$.

Выход. «Отношение является транзитивным» или «Отношение не является транзитивным».

1. Цикл по k от 1 до N, цикл по i от 1 до N, цикл по j от 1 до N.
2. Если $M_{k,i} = M_{i,j} = 1$ и $M_{k,i} = 0$, то ответ «Отношение не является транзитивным».
3. Если циклы завершены, то ответ «Отношение не является транзитивным»

Трудоемкость алгоритма $O(N^3)$.

2.2 Классификация бинарных отношений

Таким образом, в зависимости от свойств, которыми заданное бинарное отношение обладает, его можно отнести к определенному классу: **квазипорядка, эквивалентности** или **частичного порядка**.

2.2.1 Определения классов бинарных отношений

Отношение *эквивалентности* – это такое бинарное отношение между элементами множества A, для которого выполнены свойства рефлексивности, симметричности и транзитивности.

Отношение *квазипорядка* – это такое бинарное отношение, между элементами множества A, для которого выполнены свойства рефлексивности и транзитивности.

Отношение *частичного порядка* – это такое бинарное отношение, между элементами множества A, для которого выполнены свойства рефлексивности и антисимметричности.

2.2.2 Алгоритм проверки отношения на квазипорядок

Вход. Матрица $M(\rho)_{ij}$ бинарного отношения ρ размерности $N \times N$.

Выход. «Отношение является отношением квазипорядка» или «Отношение не является отношением квазипорядка».

1. Запустить проверку свойств рефлексивности и транзитивности и выполнить логическую операцию & для их результатов.
2. Если значение *Истина*, то ответ «Отношение является отношением квазипорядка».

3. Если значение *Ложь*, то ответ «Отношение не является отношением квазипорядка».

Трудоемкость алгоритма $O(N + N^3) = O(N^3)$ в силу запуска алгоритмов проверки свойств рефлексивности и транзитивности.

2.2.3 Алгоритм проверки отношения на эквивалентность

Вход. Матрица $M(\rho)_{ij}$ бинарного отношения ρ размерности $N \times N$.

Выход. «Отношение является отношением эквивалентности» или «Отношение не является отношением эквивалентности».

1. Запустить проверку свойств рефлексивности, транзитивности и симметричности и выполнить операцию & для их результатов.
2. Если получившееся значение истинно, то ответ «Отношение является отношением эквивалентности».
3. Если же значение ложно, то ответ «Отношение не является отношением эквивалентности».

Трудоемкость алгоритма $O(N + N^3 + N^2) = O(N^3)$ в силу запуска алгоритмов проверки свойств рефлексивности, транзитивности и симметричности.

2.2.4 Алгоритм проверки отношения на частичный порядок

Вход. Матрица $M(\rho)_{ij}$ бинарного отношения ρ размерности $N \times N$.

Выход. «Отношение является отношением частичного порядка» или «Отношение не является отношением частичного порядка».

1. Запустить проверку свойств рефлексивности, транзитивности и антисимметричности и выполнить операцию & для их результатов.
2. Если получившееся значение истинно, то ответ «Отношение является отношением частичного порядка».
3. Если же получившееся значение ложно, то ответ «Отношение является отношением частичного порядка».

Трудоемкость алгоритма $O(N + N^3 + N^2) = O(N^3)$ в силу запуска алгоритмов проверки свойств рефлексивности, транзитивности и симметричности.

2.3 Замыкания бинарных отношений и алгоритмы их построения

2.3.1 Определение замыканий отношения

Замыканием отношения R относительно свойства P называется такое множество R^*, что:

1. $R \subset R^*$.
2. R^* Обладает свойством P.
3. R^* является подмножеством любого другого отношения, содержащего R и обладающего свойством P.

   То есть R^* является минимальным надмножеством множества R, выдерживается P.

   Итак, исходя из вышесказанного, можно сделать вывод, что существуют 4 вида замыканий отношений: **транзитивное, симметричное, рефлексивное и эквивалентное**.

На множестве $P(A^2)$ всех бинарных отношений между элементами множества A следующие отображения являются операторами замыканий:

1. $f_r(\rho) = \varrho \cup \triangle_A$ – наименьшее рефлексивное бинарное отношение, содержащее отношение $\rho \subset A^2$.
2. $f_s(\rho) = \varrho \cup \varrho^{-1}$ – наименьшее симметричное бинарное отношение, содержащее отношение $\rho \subset A^2$.
3. $f_t(\rho) = \cup_{n=1}^{\infty} \rho^n$ – наименьшее транзитивное бинарное отношение, содержащее отношение $\rho \subset A^2$.
4. $f_{eq}(\rho) = f_t f_s f_r(\rho)$ – наименьшее отношение эквивалентности, содержащее отношение $\rho \subset A^2$.

2.3.2 Пример построения замыканий бинарного отношения

Рассмотрим множество $A = \{1,2,3,4\}$, на котором задано отношение $R =$ (1,2),(3,4),(4,2)

1. Замыканием R относительно свойства **рефлексивности**:

$$R^* = \{(1,2),(3,4),(4,2);(1,1),(2,2),(3,3),(4,4)\}$$

2. Замыканием R относительно свойства **симметричности**:

$$R^* = \{(1,2),(3,4),(4,2);(2,1),(2,4),(4,3)\}$$

3. Замыканием R относительно свойства **транзитивности**:

$$R^* = \{(1,2),(3,4),(4,2);(3,2)\}$$

2.3.3 Построение замыканий отношения

Выполним построение замыкания отношения относительно свойства рефлексивности.

Для этого выполним циклический обход по всем элементам главной диагонали матрицы отношения M_{ij} и будем проверять, находится ли элемент (M_{ii}, M_{ii}) в исходном отношении.

Вход. Матрица $M(\rho)_{ij}$ бинарного отношения ρ размерности $N \times N$.

Выход. Замыкание относительно свойства рефлексивности.

1. Создать пустой список для хранения пар замыкания, а также использовать глобальное множество для хранения пар замыкания относительно эквивалентности.
2. Цикл по i от 1 до N.
3. Если $M_{ii} = 0$, пару (i, i) добавить в замыкание рефлексивности и замыкание эквивалентности.
4. Ответ - исходное бинарное отношение и пары замыкания.

Трудоемкость алгоритма $O(N)$

Выполним построение замыкания отношения относительно свойства симметричности.

Для этого выполним циклический обход по всем элементам матрицы отношения M_{ij} и будем проверять, равны ли между собой элементы M_{ij}, M_{ji} в исходном отношении.

Вход. Матрица $M(\rho)_{ij}$ бинарного отношения ρ размерности $N \times N$.

Выход. Замыкание относительно свойства симметричности.

1. Создать пустой список для хранения пар замыкания, а также использовать глобальное множество для хранения пар замыкания относительно эквивалентности.
2. Цикл по i от 1 до N, цикл по j от 1 до N.
3. Если $M_{ij} = 1$ и $M_{ji} = 0$, добавить пару (j, i) в замыкание симметричности и замыкание эквивалентности.
4. Ответ - исходное бинарное отношение и пары замыкания.

Трудоемкость алгоритма $O(N^2)$

Выполним построение замыкания отношения относительно свойства транзитивности.

Выполним циклический обход по всем элементам матрицы M_{ij} со всеми фиксированными элементами M_{ki}:

1. Создать пустой список для хранения пар замыкания, а также использовать глобальное множество для хранения пар замыкания относительно

эквивалентности.

2. Цикл по k от 1 до N, цикл по i от 1 до N, цикл по j от 1 до N.
3. Если $M_{ki} = M_{i,j} = 1$ и $M_{ki} = 0$, то добавить пару (k, k) в замыкание транзитивности и замыкание эквивалентности.
4. Ответ - исходное бинарное отношение и пары замыкания.

Трудоемкость алгоритма $O(N^3)$

Выполним построение замыкания отношения относительно эквивалентности.

1. По очереди вызвать алгоритмы построения замыканий относительно рефлексивности, симметричности и транзитивности.
2. Ответ - исходное бинарное отношение и пары замыкания из глобальной переменной.

Трудоемкость алгоритма $O(N + N^3 + N^2) = O(N^3)$ в силу вызова алгоритмов построения замыканий относительно рефлексивности, симметричности и транзитивности

3 Программная реализация рассмотренных алгоритмов

3.1 Результаты тестирования программы

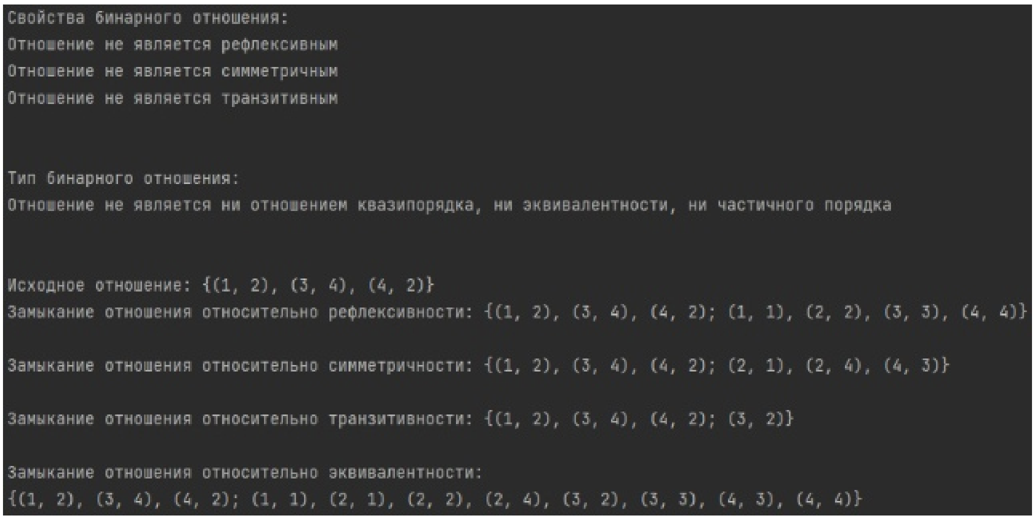

Рисунок 1

3.2 Код программы, реализующей рассмотренные алгоритмы

```python
list_for_equivalent_closure = set()


def make_set(matrix, size):

    set_view = []

    for i in range(size):
        for j in range(size):
            if matrix[i][j] == 1:
                set_view.append((i + 1, j + 1))
    return sorted(set_view)


def matrix_set_view(matrix_set, flag=None):
    if not flag:
        print('Исходное отношение: {', end='')
        print(*matrix_set, sep=', ', end='}\n')
    else:
        print('{', end='')
        print(*matrix_set, sep=', ', end='; ')


```

```
def make_reflexive(matrix, size):

    m_reflexive = []

    for i in range(size):
        if matrix[i][i] == 0:
            m_reflexive.append((i + 1, i + 1))
            list_for_equivalent_closure.add((i + 1, i + 1))

    print(*sorted(m_reflexive), sep=', ', end='}\n\n')


def make_symmetric(matrix, size):

    m_symmetric = []

    for i in range(size):
        for j in range(size):
            if matrix[i][j] == 1 and matrix[j][i] == 0:
                m_symmetric.append((j + 1, i + 1))
                list_for_equivalent_closure.add((j + 1, i + 1))

    print(*sorted(m_symmetric), sep=', ', end='}\n\n')


def make_transitive(matrix, size):

    m_transitive = []

    for k in range(size):
        for i in range(size):
            for j in range(size):
                if matrix[k][i] == matrix[i][j] == 1 and matrix[k][j] == 0:
                    m_transitive.append((k + 1, j + 1))
                    list_for_equivalent_closure.add((k + 1, j + 1))

    print(*sorted(m_transitive), sep=', ', end='}\n\n')


def is_transitive(matrix, size):

```

```
    for k in range(size):
        for i in range(size):
            for j in range(size):
                if matrix[k][i] == matrix[i][j] == 1 and matrix[k][j] == 0:
                    return False
    return True


def is_symmetric_or_antisymmetric(matrix, size):

    flag_symmetric = True
    flag_antisymmetric = True

    for i in range(size):
        for j in range(size):
            if not matrix[i][j] == matrix[j][i]:
                flag_symmetric = False
            elif matrix[i][j] == matrix[j][i] and not i == j:
                flag_antisymmetric = False
            elif not flag_symmetric and not flag_antisymmetric:
                return False, False

    return flag_symmetric, flag_antisymmetric


def is_reflexive_or_anti_reflexive(matrix, size):

    flag_reflexive = True
    flag_anti_reflexive = True

    for i in range(size):
        if matrix[i][i] == 0:
            flag_reflexive = False
        elif matrix[i][i] == 1:
            flag_anti_reflexive = False
        if not flag_reflexive and not flag_anti_reflexive:
            return False, False

    return flag_reflexive, flag_anti_reflexive


```

```python
def get_data():
    n = int(input())
    m = [[int(elem) for elem in input().split()] for _ in range(n)]
    m_set = [(i + 1, j + 1) for i in range(n) for j in range(n) if m[i][j] == 1]
    return m, sorted(m_set), n


matrix, matrix_set, size = get_data()

print('Свойства бинарного отношения:')

reflexive, anti_reflexive = is_reflexive_or_anti_reflexive(matrix, size)
if reflexive:
    print('Отношение является рефлексивным')
elif not reflexive:
    print('Отношение не является рефлексивным')
elif anti_reflexive:
    print('Отношение является антирефлексивным')
else:
    print('Отношение не является антирефлексивным')


symmetric, antisymmetric = is_symmetric_or_antisymmetric(matrix, size)
if symmetric:
    print('Отношение является симметричным')
elif not symmetric:
    print('Отношение не является симметричным')
elif antisymmetric:
    print('Отношение является антисимметричным')
else:
    print('Отношение не является антисимметричным')


transitive = is_transitive(matrix, size)
if transitive:
    print('Отношение является транзитивным')
else:
    print('Отношение не является транзитивным')

print('\n')
```

```
print('Тип бинарного отношения:')

quasi_order = True if transitive and reflexive else False
if quasi_order:
    print('Отношение является отношением квазипорядка')
    if quasi_order and symmetric:
        print('Отношение является отношением эквивалентности')
    if quasi_order and antisymmetric:
        print('Отношение является отношением частичного порядка')
else:
    print('Отношение не является ни отношением квазипорядка, ни эквивалентности, ни частичного порядка')
print('\n')

matrix_set_view(matrix_set)
if not reflexive:
    print('Замыкание отношения относительно рефлексивности: ', end='')
    matrix_set_view(matrix_set, 1)
    make_reflexive(matrix, size)

if not symmetric:
    print('Замыкание отношения относительно симметричности: ', end='')
    matrix_set_view(matrix_set, 1)
    make_symmetric(matrix, size)

if not transitive:
    print('Замыкание отношения относительно транзитивности: ', end='')
    matrix_set_view(matrix_set, 1)
    make_transitive(matrix, size)

print('Замыкание отношения относительно эквивалентности: ', end='')
matrix_set_view(matrix_set, 1)
print(*sorted(list_for_equivalent_closure), sep=', ', end='}\n\n')
'''
Примеры входных данных:

4
0 1 1 0
1 1 1 0
0 1 1 0
0 0 0 1
```

```

4
0 1 0 0
0 0 0 0
0 0 0 1
0 1 0 0
'''
```

ЗАКЛЮЧЕНИЕ

В ходе лабораторной работы были рассмотрены основные свойства бинарных отношений: рефлексивность, антирефлексивность, симметричность, антисимметричность и транзитивность. По определенным комбинациям свойств отношений, их можно классифицировать, как отношения квазипорядка (если отношение обладает свойствами транзитивности и рефлексивности), эквивалентности (если отношение является отношением квазипорядка, а также имеет свойство симметричности), а также отношения частичного порядка (если отношение является отношением квазипорядка и имеет свойство антисимметричности). Также были разработаны алгоритмы определения свойств отношений и их классификации. В ходе работы стало понятно, что самым ресурсоемким стал алгоритм определения транзитивности отношения, так как его реализация включает в себя тройной вложенный цикл.