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 2021

СОДЕРЖАНИЕ

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 соотвествущих множеств AиB по следующим формулам:

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

2.1.2 Основные свойства бинарных и алгоритмы их определения

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

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

Далее в работе представлена программная реализация определения свойств бинарных отношений.

Итак, чтобы задать матрицу M_ρ бинарного отношения ρ размерности $N \times N$, необходимо:

1. Определить мощность бинарного отношения, то есть количество элементов в строках и столбцах матрицы.
2. Если упорядоченная пара (a, b) присутствует в отношении, то $M_{i-1,j-1} = 1$, иначе $M_{i-1,j-1} = 0$

Выполним проверку свойства **рефлексивности**:

1. В полученной матрице элементы отношения ρ будут располагаться на главной диагонали, поэтому далее требуется циклически проверить, какие числа стоят на главной диагонали матрицы ($M_{a-1,a-1}$).

2. Если $M_{ij} = 1$, то цикл продолжается, если же $M_{ij} = 0$, то происходит выход из цикла и алгоритм определения рефлексивности завершается со значением False.
3. Если же цикл завершился, то алгоритм определения рефлексивности завершается со значением True.

Аналогично можно проверить свойство **антирефлексивности**.

Выполним проверку свойства свойства **симметричности**:

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

Выполним циклический проход по всем элементам матрицы отношения M_{ij}.

1. Если $M_{ij} = M_{jj}$, то циклический обход матрицы продолжается
2. Иначе роисходит выход из цикла и алгоритм определения симметричности завершается со значением False.
3. Если же циклический обход завершен, то алгоритм определения симметричности завершается со значением True.

Аналогично можно проверить свойство **антисимметричности**.

Выполним проверку свойства 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 $M_{k,j} = 1$.

Выполним циклический проход по всем элементам матрицы M_{ij} со всеми фиксирванными элементами $M_{k,i} = 1$:

1. Если $M_{k,i} = M_{i,j} = M_{k,i} = 1$, то обход матрицы продолжается.
2. Если же условие не выполнено, то происходит выход из цикла и алгоритм определения транзитивности завершается со значением False.
3. Если же циклический обход завершен, то алгоритм определения транзитивности завершается со значением True.

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**.

Итак, определим классификацию отношений путем комбинации их свойств:

1. Если $(a, b) \in \rho$ и $(b, c) \in \rho \Rightarrow (a, c) \in \rho$ и $(a, a) \notin \rho\ \forall a \in A$ (то есть отношение обладает свойствами транзитивности и рефлексивности), то данное отношение является отношением квазипорядка.

   Определим, является ли отношение отношением квазипорядка, для этого запустим вышеизложенные алгоритмы опреления свойств транзитивности и рефлексивности и сохраним результат в логическую переменную:

   а) Если значение логической переменной стало истинно, то отношение является отношением квазипорядка.

   б) Если же значение логической переменной стало ложно, то отношение не является ни отношением квазипорядка, ни отношением эквивалентности, ни отношением частичного порядка.

2. Если отношение является отношением квазипорядка и для него также выполняется свойство $(a, b) \in \rho \Rightarrow (b, a) \in \rho$ (симметричности), то данное отношение является отношением эквивалентности.

   Определим, является ли отношение отношением эквивалентности. Для этого требуется, чтобы отношение было отношением квазипорядка, а также имело свойство симметричности, поэтому запустим алгоритм определения свойства симметричности и выполним логическую операцию & с результатом предыдущего алгоритма:

   а) Если получившееся значение истинно, то отношение является отношением эквивалентности.

   б) Если же значение ложно, то отношение не является отношением эквивалентности.

3. Если же отношение является отношением квазипорядка, но обладает свойством $(a, b) \in \rho$ и $(b, a) \in \rho \Rightarrow a = b$ (антисимметричности), то данное отношение является отношением частичного порядка.

   Аналогично определим, является ли отношение отношением частичного порядка:

   а) Применим логическую операцию & для результата определения квазипорядка и результата алгоритма проверки отношения на антисимметричность.

   б) Если получившееся значение истинно, то отношение является отношением частичного порядка.

в) Если же получившееся значение ложно, то отношение не является отношением частичного порядка.

2.3 Замыкания бинарных отношений и алгоритмы их построения

2.3.1 Определение замыкания 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, обладающего свойством P.

Итак, исходя из вышесказанного, можно сделать вывод, что существуют 3 вида замыканий отношений: **транзитивное, симметричное и рефлексивное**

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

1. Замыканием R относительно свойства **рефлексивности** является
$$R^* = \{(1,2),(3,4),(4,2);(1,1),(2,2),(3,3),(4,4)\}$$
2. Замыканием R относительно свойства **симметричности** является
$$R^* = \{(1,2),(3,4),(4,2);(2,1),(2,4),(4,3)\}$$
3. Замыканием R относительно свойства **транзитивности** является
$$R^* = \{(1,2),(3,4),(4,2);(3,2)\}$$

2.3.2 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.

1. Если $M_{ii} = 0$, пара $(i+1, i+1)$ будет находиться в замыкании

2. Иначе переходим к следующему элементу на главной диагонали

Выполним построение замыкания отношения относительно свойства симметричности.

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

1. Если $M_{ij} = 1$ и $M_{ji} = 0$, пара $(j + 1, i + 1)$ будет находиться в замыкании.
2. Иначе переходим к следующему элементу

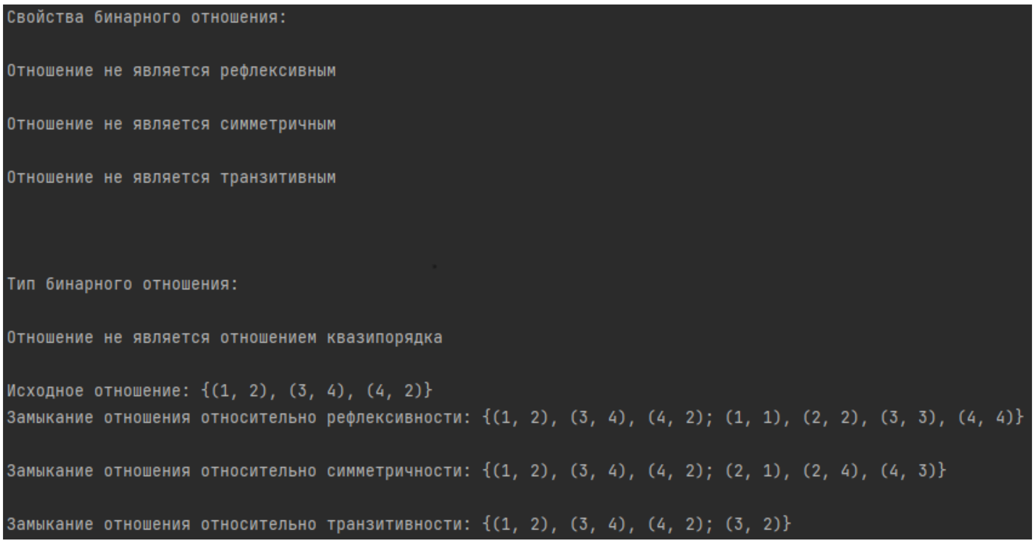
Выполним построение замыкания отношения относительно свойства транзитивности.

Выполним циклический обход по всем элементам матрицы M_{ij} со всеми фиксирванными элементами $M_{ki} = 1$:

1. Если $M_{ki} = M_{i,j} = 1$ и $M_{ki} = 0$, то пара $(k + 1, k + 1)$ будет находиться в замыкании.
2. Иначе переходим к следующей итерации обхода матрицы.

3 Программная реализация расмотренных алгоритмов

3.1 Результаты тестирования программы



```
Свойства бинарного отношения:

Отношение не является рефлексивным

Отношение не является симметричным

Отношение не является транзитивным



Тип бинарного отношения:

Отношение не является отношением квазипорядка

Исходное отношение: {(1, 2), (3, 4), (4, 2)}
Замыкание отношения относительно рефлексивности: {(1, 2), (3, 4), (4, 2); (1, 1), (2, 2), (3, 3), (4, 4)}

Замыкание отношения относительно симметричности: {(1, 2), (3, 4), (4, 2); (2, 1), (2, 4), (4, 3)}

Замыкание отношения относительно транзитивности: {(1, 2), (3, 4), (4, 2); (3, 2)}
```

Рисунок 1

3.2 Код программы, реализующей рассмотренные алгоритмы

```
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))

    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))

    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))

    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

```

```python
    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


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]
```

```python
    return m, sorted(m_set), n



matrix, matrix_set, size = get_data()

print('Свойства бинарного отношения: \n')


reflexive, anti_reflexive = is_reflexive_or_anti_reflexive(matrix, size)

if reflexive:
    print('Отношение является рефлексивным \n')
elif not reflexive:
    print('Отношение не является рефлексивным \n')
elif anti_reflexive:
    print('Отношение является антирефлексивным \n')
else:
    print('Отношение не является антирефлексивным \n')


symmetric, antisymmetric = is_symmetric_or_antisymmetric(matrix, size)

if symmetric:
    print('Отношение является симметричным \n')
elif not symmetric:
    print('Отношение не является симметричным \n')
elif antisymmetric:
    print('Отношение является антисимметричным \n')
else:
    print('Отношение не является антисимметричным \n')


transitive = is_transitive(matrix, size)

if transitive:
    print('Отношение является транзитивным \n')
else:
    print('Отношение не является транзитивным \n')

```

```
print('\n')
print('Тип бинарного отношения: \n')

quasi_order = True if transitive and reflexive else False

if quasi_order:
    print('Отношение является отношением квазипорядка \n')
    if quasi_order and symmetric:
        print('Отношение является отношением эквивалентности \n')
    if quasi_order and antisymmetric:
        print('Отношение является отношением частичного порядка \n')
else:
    print('Отношение не является отношением квазипорядка \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)

'''
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
'''
```

3.3 Оценка сложности реализованных алгоритмов в программе

3.3.1 Алгоритм проверки рефлексивности и антирефлексивности

Как было сказано в теоретической части лабораторной работы, для проверки отношения на рефлексивность, требуется один проход по главной диагонали матрицы отношения, для чего требуется $O(N)$ операций.

3.3.2 Алгоритмы проверки рефлексивности и антирефлексивности

Как было сказано в теоретической части лабораторной работы, для проверки отношения на свойства рефлексивности и антирефлексивности,требуется один проход по главной диагонали матрицы отношения, для чего требуется $O(N)$ операций.

3.3.3 Алгоритмы проверки симметричности и антисимметричности

Для проверки отношения на свойства симметричности и антисимметричности в худшем случае требуется полный проход по всем элементам матрицы отношения, что составляет $O(N^2)$ операций.

3.3.4 Алгоритм проверки транзитивности

Для проверки отношения на свойство транзитивности требуется выполнить обход по всем элементам матрицы отношения (для чего требуется один внещний кикл и один вложенный) для всех фиксированных элементов из множества A. Поэтому асимптотическая оценка данного алгоритма равна $O(N^3)$ операций.

3.3.5 Алгоритм классификации

Время работы алгоритма классификации в худшем случае составляет $O(N^3 + N^2 + 1) = O(N^3)$.

3.3.6 Алгоритм построения рефлексивного замыкания

Время работы алгоритма составляет $O(N)$

3.3.7 Алгоритм построения симметричного замыкания

Время работы алгоритма составляет $O(N^2)$

3.3.8 Алгоритм построения транзитивного замыкания

Время работы алгоритма составляет $O(N^3)$

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.