

МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение
высшего образования

**«САРАТОВСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИМЕНИ Н.Г. ЧЕРНЫШЕВСКОГО»**

Кафедра теоретических основ
компьютерной безопасности и
криптографии

Классификация бинарных отношений и системы замыканий

ОТЧЁТ

ПО ДИСЦИПЛИНЕ

«ПРИКЛАДНАЯ УНИВЕРСАЛЬНАЯ АЛГЕБРА»

студента 3 курса 331 группы

специальности 10.05.01 Компьютерная безопасность

факультета компьютерных наук и информационных технологий

Никитина Арсения Владимировича

Преподаватель

профессор, д.ф.-м.н.

подпись, дата

В. А. Молчанов

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 | 9 |
| 2.3.1 Определение замыканий отношения | 9 |
| 2.3.2 Пример построения замыканий бинарного отношения | 9 |
| 2.3.3 Алгоритм построения рефлексивного замыкания | 10 |
| 2.3.4 Алгоритм построения симметричного замыкания | 10 |
| 2.3.5 Алгоритм построения транзитивного замыкания | 10 |
| 2.3.6 Алгоритм построения эквивалентного замыкания | 11 |
| 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 соответствующих 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 $A = \{a_1, \dots, a_N\}$ мощности N . Тогда *матрицей* бинарного отношения ρ будет матрица $M(\rho)$ размерности $N \times N$, определяемая следующим образом:

$$\forall i, j = \overline{1, N} \quad M(\rho)_{ij} = \begin{cases} 1, & \text{если } (a_i, a_j) \in \rho \\ 0, & \text{если } (a_i, a_j) \notin \rho \end{cases}$$

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

Алгоритм 1. Проверка бинарного отношения на *рефлексивность*.

Вход. Матрица $M(\rho)$ 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 $M(\rho)$ 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 $M(\rho)$ 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 $M(\rho)$ 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 $M_{ij} = M_{ji} = 1$ и $j \neq i$, 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 $M(\rho)$ 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 $M_{k,j} = 0$, то ответ «Отношение не является транзитивным».
 3. Если циклы завершены, то ответ «Отношение 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, транзитивности и антисимметричности.

2.2.2 Алгоритм проверки отношения на квазипорядок

Вход. Матрица $M(\rho)$ 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 $M(\rho)$ 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 $M(\rho)$ 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 антисимметричности.

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. $f_r(\rho) = \rho \cup \Delta_A$ – наименьшее рефлексивное бинарное отношение, содержащее отношение $\rho \subset A^2$.
2. $f_s(\rho) = \rho \cup \rho^{-1}$ – наименьшее симметричное бинарное отношение, содержащее отношение $\rho \subset A^2$.
3. $f_t(\rho) = \bigcup_{n=1}^{\infty} \rho^n$ – 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 Алгоритм построения рефлексивного замыкания

Вход. Матрица $M(\rho)$ 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.
4. Ответ - замыкание бинарного отношения ρ относительно рефлексивности.

Трудоёмкость алгоритма $O(N)$

2.3.4 Алгоритм построения симметричного замыкания

Вход. Матрица $M(\rho)$ бинарного отношения ρ размерности $N \times N$.

Выход. Замыкание бинарного отношения ρ 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.
4. Ответ - замыкание бинарного отношения ρ относительно симметричности.

Трудоёмкость алгоритма $O(N^2)$

2.3.5 Алгоритм построения транзитивного замыкания

Вход. Матрица $M(\rho)$ бинарного отношения ρ размерности $N \times N$.

Выход. Замыкание бинарного отношения ρ относительно свойства транзитивности.

1. Создать копию матрицы исходного бинарного отношения.
2. Цикл по e от 1 до N , 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 $M_{kj} = 0$, то добавить пару (k, j) в замыкание транзитивности.
4. Ответ - замыкание бинарного отношения ρ относительно транзитивности.

Трудоёмкость алгоритма $O(N^4)$

2.3.6 Алгоритм построения эквивалентного замыкания

Вход. Матрица $M(\rho)$ бинарного отношения ρ размерности $N \times N$.

Выход. Эквивалентное замыкание бинарного отношения ρ .

1. По очереди вызвать алгоритмы построения замыканий относительно рефлексивности, симметричности и транзитивности.
2. Ответ - эквивалентное замыкание бинарного отношения ρ .

Трудоемкость алгоритма $O(N + N^4 + N^2) = O(N^4)$ в силу вызова алгоритмов построения замыканий относительно рефлексивности, симметричности и транзитивности

3 Программная реализация рассмотренных алгоритмов

3.1 Результаты тестирования программы

```
4
0 0 1 0
1 0 0 1
0 0 0 0
0 1 0 0
Свойства бинарного отношения:
Отношение не является рефлексивным
Отношение является антирефлексивным
Отношение не является симметричным
Отношение не является антисимметричным
Отношение не является транзитивным

Тип бинарного отношения:
Отношение не является ни отношением квазипорядка, ни эквивалентности, ни частичного порядка

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

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

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

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

Рисунок 1

3.2 Код программы, реализующей рассмотренные алгоритмы

```
1 list_for_equivalent_closure = set()
2
3
4 def make_set(matrix, size):
5
6     set_view = []
7
8     for i in range(size):
9         for j in range(size):
10             if matrix[i][j] == 1:
11                 set_view.append((i + 1, j + 1))
12     return sorted(set_view)
13
14
15 def matrix_set_view(matrix_set, flag=None):
16     if not flag:
17         print('Исходное отношение: {', end='')
18         print(*matrix_set, sep=', ', end='} \n')
19     else:
20         print('{', end='')
21         print(*matrix_set, sep=', ', end='; ')
```

```

22
23
24 def make_reflexive(matrix, size):
25
26     m_reflexive = []
27
28     for i in range(size):
29         if matrix[i][i] == 0:
30             m_reflexive.append((i + 1, i + 1))
31             list_for_equivalent_closure.add((i + 1, i + 1))
32
33     print(*sorted(m_reflexive), sep=', ', end='}\n\n')
34
35
36 def make_symmetric(matrix, size):
37
38     m_symmetric = []
39
40     for i in range(size):
41         for j in range(size):
42             if matrix[i][j] == 1 and matrix[j][i] == 0:
43                 m_symmetric.append((j + 1, i + 1))
44                 list_for_equivalent_closure.add((j + 1, i + 1))
45
46     print(*sorted(m_symmetric), sep=', ', end='}\n\n')
47
48
49 def make_transitive(copy, size):
50
51     m_transitive = []
52     for _ in range(size):
53         for k in range(size):
54             for i in range(size):
55                 for j in range(size):
56                     if copy[k][i] == copy[i][j] == 1 and copy[k][j] == 0:
57                         m_transitive.append((k + 1, j + 1))
58                         copy[k][j] = 1
59                         list_for_equivalent_closure.add((k + 1, j + 1))
60
61     print(*sorted(m_transitive), sep=', ', end='}\n\n')
62

```

```

63
64 def is_transitive(matrix, size):
65
66     for k in range(size):
67         for i in range(size):
68             for j in range(size):
69                 if matrix[k][i] == matrix[i][j] == 1 and matrix[k][j] == 0:
70                     return False
71     return True
72
73
74 def is_symmetric_or_antisymmetric(matrix, size):
75
76     flag_symmetric = True
77     flag_antisymmetric = True
78
79     for i in range(size):
80         for j in range(size):
81             if not matrix[i][j] == matrix[j][i]:
82                 flag_symmetric = False
83             if matrix[i][j] == matrix[j][i] == 1 and i != j:
84                 flag_antisymmetric = False
85             if not flag_symmetric and not flag_antisymmetric:
86                 return False, False
87
88     return flag_symmetric, flag_antisymmetric
89
90
91 def is_reflexive_or_anti_reflexive(matrix, size):
92
93     flag_reflexive = True
94     flag_anti_reflexive = True
95
96     for i in range(size):
97         if matrix[i][i] == 0:
98             flag_reflexive = False
99         elif matrix[i][i] == 1:
100             flag_anti_reflexive = False
101         if not flag_reflexive and not flag_anti_reflexive:
102             return False, False
103

```

```

104     return flag_reflexive, flag_anti_reflexive
105
106
107 def get_data():
108     n = int(input())
109     m = [[int(elem) for elem in input().split()] for _ in range(n)]
110     m_set = [(i + 1, j + 1) for i in range(n) for j in range(n) if m[i][j] ==
        ↪ 1]
111     return m, sorted(m_set), n
112
113
114 matrix, matrix_set, size = get_data()
115
116 print('Свойства бинарного отношения:')
117
118 reflexive, anti_reflexive = is_reflexive_or_anti_reflexive(matrix, size)
119 if reflexive:
120     print('Отношение является рефлексивным')
121 elif not reflexive:
122     print('Отношение не является рефлексивным')
123 if anti_reflexive:
124     print('Отношение является антирефлексивным')
125 else:
126     print('Отношение не является антирефлексивным')
127
128
129 symmetric, antisymmetric = is_symmetric_or_antisymmetric(matrix, size)
130 if symmetric:
131     print('Отношение является симметричным')
132 elif not symmetric:
133     print('Отношение не является симметричным')
134 if antisymmetric:
135     print('Отношение является антисимметричным')
136 else:
137     print('Отношение не является антисимметричным')
138
139
140 transitive = is_transitive(matrix, size)
141 if transitive:
142     print('Отношение является транзитивным')
143 else:

```

```

144     print('Отношение не является транзитивным')
145
146     print('\n ')
147     print('Тип бинарного отношения:')
148
149     quasi_order = True if transitive and reflexive else False
150     if quasi_order:
151         print('Отношение является отношением квазипорядка')
152         if quasi_order and symmetric:
153             print('Отношение является отношением эквивалентности')
154         if quasi_order and antisymmetric:
155             print('Отношение является отношением частичного порядка')
156     else:
157         print('Отношение не является ни отношением квазипорядка, ни
            ↳ эквивалентности, ни частичного порядка')
158     print('\n ')
159
160     matrix_set_view(matrix_set)
161     if not reflexive:
162         print('Замыкание отношения относительно рефлексивности: ', end='')
163         matrix_set_view(matrix_set, 1)
164         make_reflexive(matrix, size)
165
166     if not symmetric:
167         print('Замыкание отношения относительно симметричности: ', end='')
168         matrix_set_view(matrix_set, 1)
169         make_symmetric(matrix, size)
170
171     if not transitive:
172         copy = matrix
173         print('Замыкание отношения относительно транзитивности: ', end='')
174         matrix_set_view(matrix_set, 1)
175         make_transitive(copy, size)
176
177
178     print('Замыкание отношения относительно эквивалентности: ')
179     matrix_set_view(matrix_set, 1)
180     print(*sorted(list_for_equivalent_closure), sep=', ', end='} \n\n ')
181
182     '''
183     Примеры входных данных:

```


| | |
|-----|-----------|
| 184 | |
| 185 | # 4 |
| 186 | # 0 1 1 0 |
| 187 | # 1 1 1 0 |
| 188 | # 0 1 1 0 |
| 189 | # 0 0 0 1 |
| 190 | |
| 191 | # 4 |
| 192 | # 0 1 0 0 |
| 193 | # 0 0 0 0 |
| 194 | # 0 0 0 1 |
| 195 | # 0 1 0 0 |
| 196 | |
| 197 | # 3 |
| 198 | # 0 1 0 |
| 199 | # 0 0 1 |
| 200 | # 1 0 0 |
| 201 | ''' |

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.