

МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение

высшего образования

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

Кафедра теории функций и стохастического анализа

ОТЧЕТ ПО ПРОИЗВОДСТВЕННОЙ (БАЗОВОЙ) ПРАКТИКЕ

студента 4 курса 451 группы

направления 38.03.05 — Бизнес-информатика

механико-математического факультета

Чайковского Петра Ильича

Место прохождения: завод "Тантал"

Сроки прохождения: с 29.06.2019 г. по 26.07.2019 г.

Оценка:

Руководитель практики от СГУ

доцент, к. ф.-м. н.

Н. Ю. Агафонова

Руководитель практики от организации

ведущий программист

Д. Э. Кнутов

Саратов 2019

Тема практики: «Правила оформления курсовых и дипломных работ»

СОДЕРЖАНИЕ

1	Постановка задачи	4
2	Теоретические сведения по рассмотренным темам с их обоснованием .	5
3	Результаты работы	8
3.1	Алгоритм определения свойства рефлексивности.....	8
3.2	Алгоритм определения свойства симметричности.	12
3.3	Алгоритм определения свойства антисимметричности.....	16
3.4	Алгоритм определения свойства транзитивности.	20
3.5	Алгоритм классификации бинарных отношений.....	24
3.6	Алгоритм построения рефлексивного замыкания бинарного от- ношения.....	31
3.7	Алгоритм построения симметричного замыкания бинарного от- ношения.....	34
	ЗАКЛЮЧЕНИЕ	39

1 Постановка задачи

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

Порядок выполнения работы:

1. Разобрать основные определения видов бинарных отношений и разработать алгоритмы 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 по рассмотренным темам с их обоснованием

Определение. Подмножества декартова произведения $A \times B$ множеств A и B называется *бинарными отношениями* между элементами множеств A, B и обозначаются строчными греческими буквами: $\rho, \sigma, \dots, \rho_1, \rho_2, \dots$.

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

1. *рефлексивным*, если $(a, a) \in \rho$ для всякого $a \in A$;
2. *симметричным*, если $(a, b) \in \rho \implies (b, a) \in \rho$;
3. *антисимметричным*, если $(a, b) \in \rho$ и $(b, a) \in \rho \implies a = b$;
4. *транзитивным*, если $(a, b) \in \rho$ и $(b, c) \in \rho \implies (a, c) \in \rho$.

Символом Δ_A обозначается тождественное отношение на множестве A , которое определяется по формуле:

$$\Delta_A = \{ (a, a) \mid a \in A \}.$$

Тогда бинарное отношение $\rho \subset A \times A$ является:

1. *рефлексивным*, если $\Delta_A \subset \rho$;
2. *симметричным*, если $\rho^{-1} \subset \rho$;
3. *антисимметричным*, если $\rho \cap \rho^{-1} \subset \Delta_A$;
4. *транзитивным*, если $\rho\rho \subset \rho$.

Определение. Бинарное отношение ρ на множестве A называется:

1. *отношением эквивалентности (эквивалентностью)*, если оно рефлексивно, симметрично и транзитивно.
2. *отношением порядка (порядком)*, если оно рефлексивно, антисимметрично и транзитивно.
3. *отношением квазипорядка (квазипорядком)*, если оно рефлексивно и транзитивно.

Определение. Множество Z подмножеств множества A называется *системой замыканий*, если оно замкнуто относительно пересечений, т.е. выполняется:

$$\bigcap B \in Z \quad \forall B \subset Z.$$

В частности, для $\emptyset \subset Z$ выполняется $\bigcap \emptyset = A \in Z$.

Лемма 1. О системах замыканий бинарных отношений. На множестве $P(A^2)$ всех бинарных отношений между элементами множества A следующие множества являются системами замыканий:

1. Z_r - множество всех рефлексивных бинарных отношений между элементами множества A .
2. Z_s - множество всех симметричных бинарных отношений между элементами множества A .
3. Z_t - множество всех транзитивных бинарных отношений между элементами множества A .
4. $Z_{eq} = Eq(A)$ - множество всех отношений эквивалентности на множестве A .

Множество Z_{as} всех антисимметричных бинарных отношений между элементами множества A не является системой замыканий.

Определение. *Оператором замыкания* на множестве A называется отображение f множества всех подмножеств $P(A)$ в себя, удовлетворяющее условиям:

- 1) $X \subset Y \implies f(X) \subset f(Y)$;
- 2) $X \subset f(X)$;
- 3) $(f \circ f)(X) = f(X)$

для всех $X, Y \in P(A)$. Для подмножества $X \subset A$ значение $f(X)$ называется *замыканием* подмножества X .

Лемма 2. О замыканиях бинарных отношений. 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) $f_{eq}(\rho) = (f_t \circ f_s \circ f_r)(\rho)$ - наименьшее отношение эквивалентности, содержащее отношение $\rho \subset A^2$.

3 Результаты работы

3.1 Алгоритм определения свойства рефлексивности.

Описание алгоритма определения свойства рефлексивности.

Вход: список смежности бинарного отношения ρ .

Выход: строка «Бинарное отношение является/не является рефлексивным.» и bool значение **true** или **false**.

Метод: для каждого элемента a , находящегося в бинарном отношении ρ (с некоторыми элементами b_i), просматривается его список смежности. В этом списке смежности ищется сам элемент a . Алгоритм прекращает свою работу, если был найден элемент a , список смежности которого не содержит a , или, если для всякого элемента a его список смежности содержит a .

Псевдокод алгоритма определения свойства рефлексивности.

```
1 isReflexive(binaryRelation)
2 {
3     for element in binaryRelation
4     {
5         flag = false;
6
7         for subelement in element:
8             if (subelement == element):
9                 flag = true;
10
11         if (!flag)
12             return false;
13     }
14
15     return true;
16 }
```

Листинг 1: Псевдокод алгоритма.

Код программы, реализующей алгоритм определения свойства рефлексивности.

```
1 bool isReflexive(map<int, set<int>> binaryRelation)
2 {
3     bool isReflexive = false;
4     set<int> ::iterator it;
5     int i;
6
7     for (auto element : binaryRelation)
8     {
9         isReflexive = false;
10        it = element.second.begin();
11
12        for (; it != element.second.end(); ++it)
13            if (element.first == *it)
14            {
15                isReflexive = !isReflexive;
16                break;
17            }
18
19        if (!isReflexive)
20            break;
21    }
22
23    cout << "\n" <<
24        (isReflexive ? "\nBINARY RELATION IS REFLEXIVE.\n" :
25            "\nBINARY RELATION IS NOT REFLEXIVE.\n");
26    return isReflexive;
27 }
```

Листинг 2: Код программы.

Результат тестирования программы определения свойства рефлексивности.

Для демонстрации работы программы рассмотрим два произвольных бинарных отношения ρ и δ , для одного из которых свойство рефлексивности выполняется, а для другого нет.

Сгенерируем 8 пар элементов, соответствующих бинарному отношению ρ . Получаем следующие пары:

```
INPUT THE NUMBER OF PAIRS:
8
NOW INPUT THE PAIRS:
(13; 1). (16; 68). (17; 9). (34; 98). (56; 5). (60; 42). (81; 61). (91; 57).
```

Рисунок 1 – Пары элементов, находящихся в нерефлексивном бинарном отношении.

Как видно, данное отношение не является рефлексивным. Посмотрим на выход программы:

```
INPUT THE NUMBER OF PAIRS:
8
NOW INPUT THE PAIRS:
(13; 1). (16; 68). (17; 9). (34; 98). (56; 5). (60; 42). (81; 61). (91; 57).
BINARY RELATION IS NOT REFLEXIVE.
```

Рисунок 2 – Проверка на рефлексивность не пройдена.

Выход программы совпадает с тем фактом, что отношение не является рефлексивным. Теперь рассмотрим бинарное отношение δ , для которого свойство рефлексивности выполняется. Рассмотрим 8 следующих пар:

```
INPUT THE NUMBER OF PAIRS:
8
NOW INPUT THE PAIRS:
(0; 0), (0; 1), (0; 2). (1; 0), (1; 1), (1; 2). (2; 2).
```

Рисунок 3 – Пары элементов, находящихся в рефлексивном бинарном отношении.

Как видно, отношение является рефлексивным. Посмотрим на выход программы:

```
INPUT THE NUMBER OF PAIRS:
8
NOW INPUT THE PAIRS:
(0; 0), (0; 1), (0; 2). (1; 0), (1; 1), (1; 2). (2; 2).
BINARY RELATION IS REFLEXIVE.
```

Рисунок 4 – Проверка на рефлексивность пройдена.

Выход программы совпадает с тем фактом, что отношение является рефлексивным.

3.2 Алгоритм определения свойства симметричности.

Описание алгоритма определения свойства симметричности.

Вход: список смежности бинарного отношения ρ .

Выход: строка «Бинарное отношение является/не является симметричным.» и bool значение **true** или **false**.

Метод: для каждого элемента a , находящегося в бинарном отношении ρ (с некоторыми элементами b_i), просматривается его список смежности. Для каждого элемента b_i из списка смежности элемента a просматривается его список смежности, и в нём ищется элемент a . Алгоритм прекращает свою работу, если был найден элемент b_i из списка смежности элемента a , список смежности которого не содержит элемента a , или, если для всякого элемента a списки смежностей элементов b_i содержат a .

Псевдокод алгоритма определения свойства симметричности.

```
1 isSymmetric(binaryRelation)
2 {
3     for element in binaryRelation
4         for subelement in element
5             {
6                 flag = false
7
8                 if element in binaryRelation[subelement]
9                     flag = true
10
11                 if (!flag)
12                     return false;
13             }
14
15     return true;
16 }
```

Листинг 3: Псевдокод алгоритма.

Код программы, реализующей алгоритм определения свойства симметричности.

```
1 bool isSymmetric(map<int, set<int>> binaryRelation)
2 {
3     bool isSymmetric = false;
4     set<int> ::iterator it, itHelper;
5
6     for (auto element : binaryRelation)
7     {
8         it = element.second.begin();
9
10        for (; it != element.second.end(); ++it)
11        {
12            isSymmetric = false;
13
14            if (!binaryRelation[*it].empty())
15            {
16                itHelper = binaryRelation[*it].begin();
17
18                for (; itHelper != binaryRelation[*it].end(); ++itHelper)
19                    if (element.first == *itHelper)
20                    {
21                        isSymmetric = !isSymmetric;
22                        break;
23                    }
24            }
25
26            if (!isSymmetric)
27            {
28                cout << "\nBINARY RELATION IS NOT SYMMETRIC.\n";
29                return false;
30            }
31        }
32    }
33
34    cout << "\nBINARY RELATION IS SYMMETRIC.\n";
35    return true;
36 }
```

Листинг 4: Код программы.

Результат тестирования программы определения свойства симметричности.

Для демонстрации работы программы рассмотрим два произвольных бинарных отношения ρ и δ , для одного из которых свойство симметричности выполняется, а для другого нет.

Сгенерируем 8 пар элементов, соответствующих бинарному отношению ρ . Получаем следующие пары:

```
INPUT THE NUMBER OF PAIRS:
8
NOW INPUT THE PAIRS:
(7; 16). (9; 77). (19; 99). (33; 14). (35; 15). (57; 82). (79; 94). (89; 17).
```

Рисунок 5 – Пары элементов, находящихся в несимметричном бинарном отношении.

Как видно, данное отношение не является симметричным. Посмотрим на выход программы:

```
INPUT THE NUMBER OF PAIRS:
8
NOW INPUT THE PAIRS:
(7; 16). (9; 77). (19; 99). (33; 14). (35; 15). (57; 82). (79; 94). (89; 17).
BINARY RELATION IS NOT SYMMETRIC.
```

Рисунок 6 – Проверка на симметричность не пройдена.

Выход программы совпадает с тем фактом, что отношение не является симметричным. Теперь рассмотрим бинарное отношение δ , для которого свойство симметричности выполняется. Рассмотрим 8 следующих пар (число сгенерированных пар элементов может быть меньше, поскольку некоторые пары при генерации могли совпасть):

```
INPUT THE NUMBER OF PAIRS:
8
NOW INPUT THE PAIRS:
(0; 0), (0; 1), (0; 2). (1; 0), (1; 1). (2; 0).
```

Рисунок 7 – Пары элементов, находящихся в симметричном бинарном отношении.

Как видно, отношение является симметричным. Посмотрим на выход программы:

```
INPUT THE NUMBER OF PAIRS:
8
NOW INPUT THE PAIRS:
(0; 0), (0; 1), (0; 2). (1; 0), (1; 1). (2; 0).
BINARY RELATION IS SYMMETRIC.
```

Рисунок 8 – Проверка на симметричность пройдена.

Выход программы совпадает с тем фактом, что отношение является симметричным.

3.3 Алгоритм определения свойства антисимметричности.

Описание алгоритма определения свойства антисимметричности. **Вход:** список смежности бинарного отношения ρ .

Выход: строка «Бинарное отношение является/не является антисимметричным.» и bool значение **true** или **false**.

Метод: для каждого элемента a , находящегося в бинарном отношении ρ (с некоторыми элементами b_i), просматривается его список смежности. Для каждого элемента b_i из списка смежности элемента a просматривается его список смежности. В этом списке смежности не должно содержаться элемента a , за исключением ситуации, когда a и b_i - один и тот же элемент, и переход из a в b_i - петля.

Псевдокод алгоритма определения свойства антисимметричности.

```
1 isAntisymmetric(binaryRelation)
2 {
3     for element in binaryRelation:
4         for subelement in element
5             {
6                 isAntisymmetric = true;
7
8                 if subelement is not element
9                     {
10                        for elt in binaryRelation[subelement]
11                            if (elt == element)
12                                isAntisymmetric = false;
13                    }
14
15                    if (!isAntisymmetric)
16                        return false;
17            }
18
19    return true;
20 }
```

Листинг 5: Псевдокод алгоритма.

Код программы, реализующей алгоритм определения свойства антисимметричности.

```
1 bool isAntisymmetric(map<int, set<int>> binaryRelation)
2 {
3     bool isAntisymmetric = false;
4     set<int> ::iterator it, itHelper;
5
6     for (auto element : binaryRelation)
7     {
8         it = element.second.begin();
9
10        for (; it != element.second.end(); ++it)
11        {
12            isAntisymmetric = true;
13
14            if (!binaryRelation[*it].empty() && *it != element.first)
15            {
16                itHelper = binaryRelation[*it].begin();
17
18                for (; itHelper != binaryRelation[*it].end(); ++itHelper)
19                    if (*itHelper == element.first)
20                        isAntisymmetric = false;
21            }
22
23            if (!isAntisymmetric)
24            {
25                cout << "\nBINARY RELATION IS NOT ANTISYMMETRIC.\n";
26                return false;
27            }
28        }
29    }
30
31    cout << "\nBINARY RELATION IS ANTISYMMETRIC.\n";
32    return true;
33 }
```

Листинг 6: Код программы.

Результат тестирования программы определения свойства антисимметричности.

Для демонстрации работы программы рассмотрим два произвольных бинарных отношения ρ и δ , для одного из которых свойство антисимметричности выполняется, а для другого нет.

Сгенерируем 8 пар элементов (число сгенерированных пар элементов может быть меньше, поскольку некоторые пары при генерации могли совпасть), соответствующих бинарному отношению ρ . Получаем следующие пары:

```
INPUT THE NUMBER OF PAIRS:
8
NOW INPUT THE PAIRS:
(0; 0), (0; 2). (1; 1). (2; 0), (2; 2).
```

Рисунок 9 – Пары элементов, находящихся в неантисимметричном бинарном отношении.

Как видно, данное отношение не является антисимметричным. Посмотрим на выход программы:

```
INPUT THE NUMBER OF PAIRS:
8
NOW INPUT THE PAIRS:
(0; 0), (0; 2). (1; 1). (2; 0), (2; 2).

BINARY RELATION IS NOT ANTISYMMETRIC.
```

Рисунок 10 – Проверка на антисимметричность не пройдена.

Выход программы совпадает с тем фактом, что отношение не является антисимметричным. Теперь рассмотрим бинарное отношение δ , для которого свойство антисимметричности выполняется. Рассмотрим 8 следующих пар:

```
INPUT THE NUMBER OF PAIRS:
8
NOW INPUT THE PAIRS:
(36; 73). (37; 52). (42; 16). (64; 53). (94; 79). (95; 73). (96; 90). (97; 13).
```

Рисунок 11 – Пары элементов, находящихся в симметричном бинарном отношении.

Как видно, отношение является антисимметричным. Посмотрим на выход программы:

```
INPUT THE NUMBER OF PAIRS:
8
NOW INPUT THE PAIRS:
(36; 73). (37; 52). (42; 16). (64; 53). (94; 79). (95; 73). (96; 90). (97; 13).
BINARY RELATION IS ANTISYMMETRIC.
```

Рисунок 12 – Проверка на симметричность пройдена.

Выход программы совпадает с тем фактом, что отношение является антисимметричным.

3.4 Алгоритм определения свойства транзитивности.

Описание алгоритма определения свойства транзитивности.

Вход: список смежности бинарного отношения ρ .

Выход: строка «Бинарное отношение является/не является транзитивным.» и bool значение **true** или **false**.

Метод: для каждого элемента a , находящегося в бинарном отношении ρ (с некоторыми элементами b_i), просматривается его список смежности. Для каждого элемента b_i из списка смежности элемента a просматривается его список смежности. Для удовлетворения условия транзитивности каждый элемент из списка смежности b_i должен содержаться в списке смежности элемента a . Алгоритм завершает свою работу, если находится такой элемент, что это условие для него не выполняется, или после проверки и выполнения этого условия для всякого элемента a .

Псевдокод алгоритма определения свойства транзитивности.

```
1 isTransitive(binaryRelation)
2 {
3     for element in binaryRelation:
4         for subelement in element:
5             for elt in binaryRelation[subelement]
6                 {
7                     isTransitive = false;
8
9                     for eltSup in element:
10                        if (elt == eltSup)
11                            isTransitive = true
12
13                if (!isTransitive)
14                    return false;
15            }
16        }
17    }
18
19    return true;
20 }
```

Листинг 7: Псевдокод алгоритма.

Код программы, реализующей алгоритм определения свойства транзитивности.

```
1 bool isTransitive(map<int, set<int>> binaryRelation)
2 {
3     bool isTransitive = false;
4     set<int> ::iterator it, itHelper, itSup;
5
6     for (auto element : binaryRelation)
7     {
8         it = element.second.begin();
9
10        for (; it != element.second.end(); ++it)
11        {
12            if (!binaryRelation[*it].empty())
13            {
14                itHelper = binaryRelation[*it].begin();
15
16                for (; itHelper != binaryRelation[*it].end(); ++itHelper)
17                {
18                    itSup = element.second.begin();
19                    isTransitive = false;
20
21                    for (; itSup != element.second.end(); ++itSup)
22                        if (*itHelper == *itSup)
23                            isTransitive = !isTransitive;
24
25                    if (!isTransitive)
26                    {
27                        cout << "\nBINARY RELATION IS NOT TRANSITIVE.\n";
28                        return false;
29                    }
30                }
31            }
32        }
33    }
34    cout << "\nBINARY RELATION IS TRANSITIVE.\n";
35    return true;
36 }
```

Листинг 8: Код программы.

Результат тестирования программы определения свойства транзитивности.

Для демонстрации работы программы рассмотрим два произвольных бинарных отношения ρ и δ , для одного из которых свойство транзитивности выполняется, а для другого нет.

Сгенерируем 8 пар элементов, соответствующих бинарному отношению ρ . Получаем следующие пары:

```
INPUT THE NUMBER OF PAIRS:
8
NOW INPUT THE PAIRS:
(6; 67). (17; 14), (17; 16). (18; 54). (19; 88). (26; 90). (90; 24). (92; 97).
```

Рисунок 13 – Пары элементов, находящихся в нетранзитивном бинарном отношении.

Как видно, данное отношение не является транзитивным. Посмотрим на выход программы:

```
INPUT THE NUMBER OF PAIRS:
8
NOW INPUT THE PAIRS:
(6; 67). (17; 14), (17; 16). (18; 54). (19; 88). (26; 90). (90; 24). (92; 97).
BINARY RELATION IS NOT TRANSITIVE.
```

Рисунок 14 – Проверка на транзитивность не пройдена.

Выход программы совпадает с тем фактом, что отношение не является транзитивным. Теперь рассмотрим бинарное отношение δ , для которого свойство транзитивности выполняется. Перейдём на ручной ввод и введём 8 пар элементов:

```
INPUT THE NUMBER OF PAIRS:
8
NOW INPUT THE PAIRS:
1 1 1 2 2 3 1 3 4 5 5 6 4 6 2 2
THE ELEMENTS OF YOUR BINARY RELATION ARE:
(1; 1), (1; 2), (1; 3). (2; 2), (2; 3). (4; 5), (4; 6). (5; 6).
```

Рисунок 15 – Пары элементов, находящихся в транзитивном бинарном отношении.

Как видно, отношение является транзитивным. Посмотрим на выход программы:

```
INPUT THE NUMBER OF PAIRS:
8
NOW INPUT THE PAIRS:
1 1 1 2 2 3 1 3 4 5 5 6 4 6 2 2

THE ELEMENTS OF YOUR BINARY RELATION ARE:
(1; 1), (1; 2), (1; 3). (2; 2), (2; 3). (4; 5), (4; 6). (5; 6).

BINARY RELATION IS TRANSITIVE.
```

Рисунок 16 – Проверка на транзитивность пройдена.

Выход программы совпадает с тем фактом, что отношение является транзитивным.

3.5 Алгоритм классификации бинарных отношений.

Отдельно рассмотрим три функции, определяющих класс бинарного отношения, а именно: отношение эквивалентности, отношение порядка или отношение квазипорядка. Тогда алгоритм, содержащий в себе эти три функции, может выглядеть следующим образом:

Вход: список смежности бинарного отношения ρ .

Выход: строки «Бинарное отношение является/не является отношением эквивалентности.», «Бинарное отношение является/не является отношением порядка.», «Бинарное отношение является/не является отношением квазипорядка.»

Метод: последовательно запускаем функции проверки принадлежности бинарного отношения к соответствующему классу.

Код программы, реализующей алгоритм классификации бинарных отношений.

```
1 void isEquivalence(map<int, set<int>> binaryRelation)
2 {
3     if (isReflexive(binaryRelation) &&
4         isSymmetric(binaryRelation) &&
5         isTransitive(binaryRelation))
6         cout << "\nBINARY RELATION IS AN EQUIVALENT RELATION.\n";
7     else
8         cout << "\nBINARY RELATION IS NOT AN EQUIVALENT RELATION.\n";
9 }
10
11 void isOrder(map<int, set<int>> binaryRelation)
12 {
13     if (isReflexive(binaryRelation) &&
14         isAntisymmetric(binaryRelation) &&
15         isTransitive(binaryRelation))
16         cout << "\nBINARY RELATION IS AN ORDER RELATION.\n";
17     else
18         cout << "\nBINARY RELATION IS NOT AN ORDER RELATION.\n";
19 }
20
21 void isQuasi(map<int, set<int>> binaryRelation)
```



```
22 {  
23     if (isReflexive(binaryRelation) &&  
24         isTransitive(binaryRelation))  
25         cout << "\nBINARY RELATION IS A QUASI-ORDER RELATION.\n";  
26     else  
27         cout << "\nBINARY RELATION IS NOT A QUASI-ORDER RELATION.\n";  
28 }
```

Листинг 9: Набор функций, осуществляющий классификацию бинарных отношений.

Результат тестирования программы классификации бинарных отношений.

Для демонстрации работы программы рассмотрим шесть произвольных бинарных отношений ρ_i , $i = \overline{1,6}$. В каждой паре одно отношение будет принадлежать какому-нибудь классу, а другое - нет. Начнём с рассмотрения отношения эквивалентности.

Сгенерируем 8 пар элементов, соответствующих бинарному отношению ρ_1 . Получаем следующие пары:

```
INPUT THE NUMBER OF PAIRS:
8
NOW INPUT THE PAIRS:
(6; 37). (23; 69). (29; 82). (31; 78). (56; 33). (60; 88). (66; 58). (91; 77).
```

Рисунок 17 – Пары элементов, находящихся в неэквивалентном бинарном отношении.

Как видно, данное отношение не является эквивалентностью. Посмотрим на выход программы:

```
INPUT THE NUMBER OF PAIRS:
8
NOW INPUT THE PAIRS:
(6; 37). (23; 69). (29; 82). (31; 78). (56; 33). (60; 88). (66; 58). (91; 77).

BINARY RELATION IS NOT REFLEXIVE.

BINARY RELATION IS NOT AN EQUIVALENT RELATION.
```

Рисунок 18 – Проверка на эквивалентность не пройдена.

Выход программы совпадает с тем фактом, что отношение не является эквивалентностью. Также программа сообщила, почему данное отношение не является эквивалентностью - оно нереклексивно. Теперь рассмотрим бинарное отношение ρ_2 , которое является эквивалентностью. Перейдём на ручной ввод и введём 9 следующих пар:

```

INPUT THE NUMBER OF PAIRS:
9
NOW INPUT THE PAIRS:
1 1 2 2 3 3 1 2 2 1 2 3 3 2 1 3 3 1

THE ELEMENTS OF YOUR BINARY RELATION ARE:
(1; 1), (1; 2), (1; 3). (2; 1), (2; 2), (2; 3). (3; 1), (3; 2), (3; 3).

```

Рисунок 19 – Пары элементов, находящихся в эквивалентном бинарном отношении.

Как видно, отношение является эквивалентностью. Посмотрим на выход программы:

```

INPUT THE NUMBER OF PAIRS:
9
NOW INPUT THE PAIRS:
1 1 2 2 3 3 1 2 2 1 2 3 3 2 1 3 3 1

THE ELEMENTS OF YOUR BINARY RELATION ARE:
(1; 1), (1; 2), (1; 3). (2; 1), (2; 2), (2; 3). (3; 1), (3; 2), (3; 3).

BINARY RELATION IS REFLEXIVE.

BINARY RELATION IS SYMMETRIC.

BINARY RELATION IS TRANSITIVE.

BINARY RELATION IS AN EQUIVALENT RELATION.

```

Рисунок 20 – Проверка на эквивалентность пройдена.

Выход программы совпадает с тем фактом, что отношение является эквивалентностью, поскольку каждое из необходимых свойств выполняется.

Теперь перейдём к рассмотрению отношения порядка.

Сгенерируем 8 пар элементов, соответствующих бинарному отношению ρ_3 . Получаем следующие пары:

```

INPUT THE NUMBER OF PAIRS:
8
NOW INPUT THE PAIRS:
(8; 38). (39; 34). (41; 84). (43; 42). (46; 54). (64; 49). (73; 30). (86; 86).

```

Рисунок 21 – Пары элементов, не находящихся в бинарном отношении порядка.

Как видно, данное отношение не является порядком. Посмотрим на выход программы:

```

INPUT THE NUMBER OF PAIRS:
8
NOW INPUT THE PAIRS:
(8; 38). (39; 34). (41; 84). (43; 42). (46; 54). (64; 49). (73; 30). (86; 86).

BINARY RELATION IS NOT REFLEXIVE.

BINARY RELATION IS NOT AN ORDER RELATION.

```

Рисунок 22 – Проверка на упорядоченность не пройдена.

Выход программы совпадает с тем фактом, что отношение не является отношением порядка. Также программа сообщила, почему данное отношение не является отношением порядка - оно нерефлексивно. Теперь рассмотрим бинарное отношение ρ_4 , которое является отношением порядка. Перейдём на ручной ввод и введём 6 следующих пар:

```

INPUT THE NUMBER OF PAIRS:
6
NOW INPUT THE PAIRS:
1 1 1 2 1 3 2 2 2 3 3 3

THE ELEMENTS OF YOUR BINARY RELATION ARE:
(1; 1), (1; 2), (1; 3). (2; 2), (2; 3). (3; 3).

```

Рисунок 23 – Пары элементов, находящиеся в упорядоченном бинарном отношении.

Как видно, отношение является отношением порядка. Посмотрим на выход программы:

```

INPUT THE NUMBER OF PAIRS:
6
NOW INPUT THE PAIRS:
1 1 1 2 1 3 2 2 2 3 3 3

THE ELEMENTS OF YOUR BINARY RELATION ARE:
(1; 1), (1; 2), (1; 3). (2; 2), (2; 3). (3; 3).

BINARY RELATION IS REFLEXIVE.

BINARY RELATION IS ANTISYMMETRIC.

BINARY RELATION IS TRANSITIVE.

BINARY RELATION IS AN ORDER RELATION.

```

Рисунок 24 – Проверка на упорядоченность пройдена.

Выход программы совпадает с тем фактом, что отношение является по-

рядком, поскольку каждое из необходимых свойств выполняется.

Теперь перейдём к рассмотрению отношения квазипорядка.

Сгенерируем 8 пар элементов, соответствующих бинарному отношению ρ_5 . Получаем следующие пары:

```
INPUT THE NUMBER OF PAIRS:
8
NOW INPUT THE PAIRS:
(11; 36). (20; 91). (23; 44). (45; 81). (63; 50). (74; 34). (81; 36). (82; 71).
```

Рисунок 25 – Пары элементов, не находящихся в бинарном отношении квазипорядка.

Как видно, данное отношение не является квазипорядком. Посмотрим на выход программы:

```
INPUT THE NUMBER OF PAIRS:
8
NOW INPUT THE PAIRS:
(11; 36). (20; 91). (23; 44). (45; 81). (63; 50). (74; 34). (81; 36). (82; 71).
BINARY RELATION IS NOT REFLEXIVE.
BINARY RELATION IS NOT A QUASI-ORDER RELATION.
```

Рисунок 26 – Проверка на упорядоченность не пройдена.

Выход программы совпадает с тем фактом, что отношение не является отношением квазипорядка. Также программа сообщила, почему данное отношение не является отношением квазипорядка - оно нереклексивно. Теперь рассмотрим бинарное отношение ρ_6 , которое является отношением квазипорядка. Перейдём на ручной ввод и введём 7 следующих пар:

```
INPUT THE NUMBER OF PAIRS:
7
NOW INPUT THE PAIRS:
1 1 1 2 1 3 2 1 2 2 2 3 3 3
THE ELEMENTS OF YOUR BINARY RELATION ARE:
(1; 1), (1; 2), (1; 3). (2; 1), (2; 2), (2; 3). (3; 3).
```

Рисунок 27 – Пары элементов, находящихся в квазиупорядоченном бинарном отношении.

Как видно, отношение является отношением квазипорядка. Посмотрим на выход программы:

```
INPUT THE NUMBER OF PAIRS:
7
NOW INPUT THE PAIRS:
1 1 1 2 1 3 2 1 2 2 2 3 3 3

THE ELEMENTS OF YOUR BINARY RELATION ARE:
(1; 1), (1; 2), (1; 3). (2; 1), (2; 2), (2; 3). (3; 3).

BINARY RELATION IS REFLEXIVE.

BINARY RELATION IS TRANSITIVE.

BINARY RELATION IS A QUASI-ORDER RELATION.
```

Рисунок 28 – Проверка на квазиупорядоченность пройдена.

Выход программы совпадает с тем фактом, что отношение является квазипорядком, поскольку каждое из необходимых свойств выполняется.

3.6 Алгоритм построения рефлексивного замыкания бинарного отношения.

Описание алгоритма построения рефлексивного замыкания бинарного отношения. **Вход:** список смежности бинарного отношения ρ .

Выход: замкнутое относительно свойства рефлексивности бинарное отношение ρ .

Метод: для каждого элемента a , находящегося в бинарном отношении ρ (с некоторыми элементами b_i), в его список смежности добавляется элемент a .

Псевдокод алгоритма определения свойства транзитивности.

```
1 reflexiveClosure(binaryRelation)
2 {
3   for element in binaryRelation
4   {
5     placeholder = element
6     binaryRelation[element].insert(element);
7   }
8
9   return binaryRelation;
10 }
```

Листинг 10: Псевдокод алгоритма.

Код программы, реализующей алгоритм построения рефлексивного замыкания бинарного отношения.

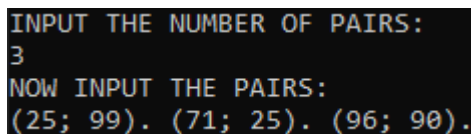
```
1 map<int, set<int>>
2     reflexiveClosure(map<int, set<int>> binaryRelation)
3 {
4     int i, placeholder;
5     set<int> ::iterator it;
6
7     for (auto element : binaryRelation)
8     {
9         placeholder = element.first;
10
11         binaryRelation[element.first].insert(element.first);
12     }
13
14     return binaryRelation;
15 }
```

Листинг 11: Код программы.

Результат тестирования программы рефлексивного замыкания бинарного отношения.

Для демонстрации работы программы рассмотрим произвольное отношение δ .

Сгенерируем 3 пары элементов, соответствующих бинарному отношению δ . Получаем следующие пары:



```
INPUT THE NUMBER OF PAIRS:
3
NOW INPUT THE PAIRS:
(25; 99). (71; 25). (96; 90).
```

Рисунок 29 – Бинарное отношение, не обладающее свойством рефлексивности.

Как видно, данное отношение не является рефлексивным. Построим рефлексивное замыкание этого бинарного отношения:

```
INPUT THE NUMBER OF PAIRS:
3
NOW INPUT THE PAIRS:
(25; 99). (71; 25). (96; 90).

THE ELEMENTS OF YOUR BINARY RELATION AFTER REFLEXIVE CLOSURE ARE:
(25; 25), (25; 99). (71; 25), (71; 71). (90; 90). (96; 90), (96; 96). (99; 99).

BINARY RELATION IS REFLEXIVE.
```

Рисунок 30 – Бинарное отношение после рефлексивного замыкания.

Как можно видеть, получившееся бинарное отношение является замкнутым относительно свойства рефлексивности, что подтверждает выход программы *isReflexive*.

3.7 Алгоритм построения симметричного замыкания бинарного отношения.

Описание алгоритма построения симметричного замыкания бинарного отношения.

Вход: список смежности бинарного отношения ρ .

Выход: замкнутое относительно свойства симметричности бинарное отношение ρ .

Метод: для каждого элемента a , находящегося в бинарном отношении ρ с некоторыми элементами b_i , в список смежности каждого такого элемента b_i добавляется элемент a .

Псевдокод алгоритма определения свойства транзитивности.

```
1 symmetricClosure(binaryRelation)
2 {
3     for element in binaryRelation:
4         for subelement in element:
5             binaryRelation[subelement].insert(element)
6
7     return binaryRelation;
8 }
```

Листинг 12: Псевдокод алгоритма.

Код программы, реализующей алгоритм построения симметричного замыкания бинарного отношения.

```
1 map<int, set<int>>
2     symmetricClosure(map<int, set<int>> binaryRelation)
3 {
4     set<int> ::iterator it;
5     for (auto element : binaryRelation)
6     {
7         it = element.second.begin();
8         for (; it != element.second.end(); ++it)
9             binaryRelation[*it].insert(element.first);
10    }
11    return binaryRelation;
12 }
```

Листинг 13: Код программы.

Результат тестирования программы симметричного замыкания бинарного отношения.

Для демонстрации работы программы рассмотрим произвольное отношение δ .

Сгенерируем 3 пары элементов, соответствующих бинарному отношению δ . Получаем следующие пары:

```
INPUT THE NUMBER OF PAIRS:
4
NOW INPUT THE PAIRS:
(7; 95). (70; 46). (89; 99). (93; 61).
```

Рисунок 31 – Бинарное отношение, не обладающее свойством симметричности.

Как видно, данное отношение не является симметричным. Построим симметричное замыкание этого бинарного отношения:

```
INPUT THE NUMBER OF PAIRS:
4
NOW INPUT THE PAIRS:
(7; 95). (70; 46). (89; 99). (93; 61).

THE ELEMENTS OF YOUR BINARY RELATION AFTER SYMMETRIC CLOSURE ARE:
(7; 95). (46; 70). (61; 93). (70; 46). (89; 99). (93; 61). (95; 7). (99; 89).

BINARY RELATION IS SYMMETRIC.
```

Рисунок 32 – Бинарное отношение после симметричного замыкания.

Как можно видеть, получившееся бинарное отношение является замкнутым относительно свойства симметричности, что подтверждает выход программы *isSymmetric*.

Описание алгоритма построения транзитивного замыкания бинарного отношения.

Вход: список смежности бинарного отношения ρ .

Выход: замкнутое относительно свойства транзитивности бинарное отношение ρ .

Метод: для каждого элемента a , находящегося в бинарном отношении ρ с некоторыми элементами b_i , все элементы, находящиеся в списке смежности элемента b_i , должны быть добавлены в список смежности элемента a . Но поскольку после всего одного применения функции *transitiveClosure* получившееся бинарное отношение может оказаться незамкнутым относительно свойства транзитивности, применение функции на ρ надо продолжать до тех пор, пока функция *isTransitive* не вернёт **true**.

Псевдокод алгоритма определения свойства транзитивности.

```
1 transitiveClosureMachinery(binaryRelation)
2 {
3     for element in binaryRelation:
4         for subelement in element:
5             for elt in binaryRelation[subelement]:
6                 binaryRelation[element].insert(elt);
7
8     return binaryRelation;
9 }
10
11 transitiveClosure(binaryRelation)
12 {
13     while (!isTransitive(binaryRelation))
14         binaryRelation =
15             transitiveClosureMachinery(binaryRelation);
16
17     return binaryRelation;
18 }
```

Листинг 14: Псевдокод алгоритма.

Код программы, реализующей алгоритм построения симметричного замыкания бинарного отношения.

```
1 map<int, set<int>>
2     transitiveClosure(map<int, set<int>> binaryRelation)
3 {
4     set<int> ::iterator it, itHelper;
5
6     for (auto element : binaryRelation)
7     {
8         it = element.second.begin();
9
10        for (; it != element.second.end(); ++it)
11        {
12            itHelper = binaryRelation[*it].begin();
13
14            for (; itHelper != binaryRelation[*it].end(); ++itHelper)
15                binaryRelation[element.first].insert(*itHelper);
16        }
17    }
18
19    return binaryRelation;
20 }
21
22 map<int, set<int>>
23     transitiveClosure(map<int, set<int>> binaryRelation)
24 {
25     while (!isTransitive(binaryRelation))
26         binaryRelation =
27             transitiveClosureMachinery(binaryRelation);
28
29     return binaryRelation;
30 }
```

Листинг 15: Код программы.

Результат тестирования программы транзитивного замыкания бинарного отношения.

Для демонстрации работы программы рассмотрим произвольное отношение δ .

Перейдём на ручной ввод и сгенерируем 6 пар элементов, соответствующих бинарному отношению δ . Получаем следующие пары:

```
INPUT THE NUMBER OF PAIRS:
6
NOW INPUT THE PAIRS:
0 2 1 4 2 2 2 4 4 3 4 4

THE ELEMENTS OF YOUR BINARY RELATION ARE:
(0; 2). (1; 4). (2; 2), (2; 4). (4; 3), (4; 4).
```

Рисунок 33 – Бинарное отношение, не обладающее свойством транзитивности.

Как видно, данное отношение не является транзитивным. Построим транзитивное замыкание этого бинарного отношения:

```
INPUT THE NUMBER OF PAIRS:
6
NOW INPUT THE PAIRS:
0 2 1 4 2 2 2 4 4 3 4 4

THE ELEMENTS OF YOUR BINARY RELATION ARE:
(0; 2). (1; 4). (2; 2), (2; 4). (4; 3), (4; 4).

BINARY RELATION IS NOT TRANSITIVE.

BINARY RELATION IS NOT TRANSITIVE.

BINARY RELATION IS TRANSITIVE.

THE ELEMENTS OF YOUR BINARY RELATION AFTER TRANSITIVE CLOSURE ARE:
(0; 2), (0; 3), (0; 4). (1; 3), (1; 4). (2; 2), (2; 3), (2; 4). (4; 3), (4; 4).
```

Рисунок 34 – Бинарное отношение после транзитивного замыкания.

Заметим, что $(f \circ f)(\delta)$ (где f - функция *transitiveClosureMachinery*) оказалось недостаточно для замыкания исходного бинарного отношения.

Получившееся бинарное отношение является замкнутым относительно свойства транзитивности, что при третьем применении функции *transitiveClosureMachinery* подтверждает выход программы *isTransitive*.

ЗАКЛЮЧЕНИЕ