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»

студентки 3 курса 331 группы
специальности 10.05.01 Компьютерная безопасность
факультета компьютерных наук и информационных технологий
Шуликиной Анастасии Александровны

Преподаватель
профессор, д.ф.-м.н. ___________ В. А. Молчанов
подпись, дата

Саратов 2022

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

В данной лабораторной работе поставлена задача рассмотрения основных свойств бинарных отношений, их классификация и замыкание, а также написание алгоритмов для опредления классификации и замыкания бинарного отношения.

1 Цель работы и порядок её 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 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 Теория

2.1 Бинарные отношения и их свойства

Бинарным отношением между элементами A и B называется любое подмножество ρ множества $A \times B$, то есть $\rho \subset A \times B$.

По определению, бинарным отношением называется множество пар. Если ρ – бинарное отношение (т.е. множество пар), то говорят, что параметры x и y связаны бинарным отношением ρ, если пара $\langle x, y \rangle$ является элементом ρ, т.е. $\langle x, y \rangle \in \rho$.

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

— рефлекисвным, если $(a, a) \in \rho \; \forall a \in A$;

— симметричным, если $(a, b) \in \rho \Rightarrow (b, a) \in \rho \; \forall a, b \in A$;

— антисимметричным, если $(a, b) \in \rho$ и $(b, a) \in \rho \Rightarrow a = b \; \forall a, b \in A$;

— транзитивным, если $(a, b) \in \rho$ и $(b, c) \in \rho \Rightarrow (a, c) \in \rho \; \forall a, b, c \in A$.

Матрица бинарного отношения ρ – это прямоугольная таблица, строки которой соответствуют первым координатам, а столбцы – вторым координатам. На пересечении i-й строки и j-ого столбца ставится 1, если выполняется соотношение aiRaj, и 0, если оно не выполняется.

2.1.1 Свойства симметричности

Симметричной называют квадратную матрицу, элементы которой симметричны относительно главной диагонали.

Алгоритм проверки бинарного отношения на симметричность:

Вход. Матрица $M(\rho)$ бинарного отношения p размерностью $N \times N$.

Выход. «Отношение симметрично» или ничего не выводится.

<u>Шаг 1.</u> Цикл по i от 1 до N, цикл по j от 1 до N.

<u>Шаг 2.</u> Если $M_{ij} = M_{ji}$, то ответ «Отношение симметрично».

Трудоемкость алгоритма $O(N^2)$

2.1.2 Свойства антисимметричности

Алгоритм проверки бинарного отношения на антисимметричность:

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

Выход. «Отношение антисимметрично» или ничего не выводится.

<u>Шаг 1.</u> Цикл по i от 1 до N, цикл по j от 1 до N.

<u>Шаг 2.</u> Если $M_{ij} = -M_{ji}$, то ответ «Отношение антисимметрично».

Трудоемкость алгоритма $O(N^2)$

2.1.3 Свойства рефлексивности

Рефлексивной называют такую матрицу, у которой все диагональные элементы равняюся 1.

Алгоритм проверки бинарного отношения на рефлексивность:

Вход. Матрица $M(\rho)$ бинарного отношения p размерностью $N \times N$.

Выход. «Отношение рефлексивно» или ничего не выводится.

Шаг 1. Цикл по i от 1 до N.

Шаг 2. Если $M_{ii} = 1$, то ответ «Отношение рефлексивно», иначе отношение не рефлексивно.

Трудоемкость алгоритма $O(N)$

2.1.4 Свойства транзитивности

Транзитивной называют матрицу, если для любого фиксированного элемента $M_{ki} = 1$ из матрицы отношения, и для любого элемента из матрицы отношения $Mij = 1$, выполняется $M_{kj} = 1$.

Алгоритм проверки бинарного отношения на транзитивность:

Вход. Матрица $M(\rho)$ бинарного отношения p размерностью $N \times N$.

Выход. «Отношение транзитивно» или ничего не выводится.

Шаг 1. Цикл по k от 1 до N, цикл по i от 1 до N, цикл по j от 1 до N.

Шаг 2. Проверяем все значение M_{ij}, если значение равно 1, то проверяем все значения M_{jk}, если значение равно 1, то проверяем значение M_{ik}, если оно равно 0, то ответ «Отношение транзитивно», иначе отношение не транзитивно.

Трудоемкость алгоритма $O(N^3)$

2.2 Типы бинарных отношений

Существует три основных типа бинарных отношений:

— отношение эквивалентности
— отношение квазипорядка
— отношение частичного порядка

2.2.1 Отношение эквивалентности

Бинарное отношение ε на множестве A называют отношением эквивалентности, если оно рефлексивно, симметрично и транзитивно.

Алгоритм проверки отношения на эквивалентность:

Вход. Матрица $M(\rho)$ бинарного отношения p размерностью $N \times N$.

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

Шаг 1. Результат = симмтеричность && рефлексивность && транзитивность, где симметричность - алгоритм 2.1.1, рефлексивность - 2.1.3, транзитивность - 2.1.4.

Шаг 2. Производится проверка на истинность или ложность. Если получившееся значение – истинно, то ответ «Отношение является отношением эквивалентности», если ложно, ответ «Отношение отношением эквивалентности не является».

Трудоемкость алгоритма $O(N^3) = O(N + N^2 + N^3)$

2.2.2 Отношение квазипорядка

Бинарное отношение ε на множестве A называют отношением квазипорядка, если оно рефлексивно и транзитивно.

Алгоритм проверки отношения на отношение квазипорядка:

Вход. Матрица $M(\rho)$ бинарного отношения p размерностью $N \times N$.

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

Шаг 1. Из определения, для того, чтобы бинарное отношение являлось отношением квазипорядка, оно должно включать в себя свойства рефлексивности и транзитивности. Поэтому первым делом производится проверка на рефлексивность и транзитивность. Выполняется операция & всех результатов.

Шаг 2. Производится проверка на истинность или ложность. Если получившееся значение – истинно, то ответ «Отношение является отношением квазипорядка», если ложно, ответ «Отношение отношением квазипорядка не является».

Трудоемкость алгоритма $O(N^3) = O(N + N^3)$

2.2.3 Отношение частичного порядка

Бинарное отношение ε на множестве A называют отношением частичного порядка, если оно рефлексивно, антисимметрично и транзитивно.

Алгоритм проверки отношения на отношение частичного порядка:

Вход. Матрица $M(\rho)$ бинарного отношения p размерностью $N \times N$.

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

Шаг 1. Из определения, для того, чтобы бинарное отношение являлось отношением частичного порядка, оно должно включать в себя свойства рефлексивности, антисимметричности и транзитивности. Поэтому первым делом производится проверка на рефлексивность, антисимметричность и транзитивность. Выполняется операция & всех результатов.

Шаг 2. Производится проверка на истинность или ложность. Если получившееся значение – истинно, то ответ «Отношение является отношением частичного порядка», если ложно, ответ «Отношение отношением частичного порядка не является».

Трудоемкость алгоритма $O(N^3) = O(N + N^2 + N^3)$

2.3 Системы замыкания на множестве бинарных отношений

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.

Множество Z подмножеств множества A называется системой замыканий, если оно замкнуто относительно пересечений, т.е. выполняется $\cap B \in Z$ для любого подмножества $B \subset Z$.

В частности, для $\varnothing \subset Z$ выполняется $\cap\varnothing = A \in Z$

На множестве всех бинарных отношений между элементами множества A^2 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 замыкания.

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

1. $f_r(\rho) = \rho \cup \triangle_A$ – 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. $f_t(\rho) = \cup_{n=1}^{\infty} \rho^n$ – наименьшее транзитивное бинарное отношение, содержащее отношение $\rho \subset A^2$,
4. $f_e q(\rho) = f_t f_s f_r(\rho)$ – наименьшее отношение эквивалентности, содержащее отношение $\rho \subset A^2$.

2.3.1 Алгоритм построения замыкания отношения относительно свойства рафлексивности

Вход. Матрица $M(\rho)$ бинарного отношения p размерностью $N \times N$.

Выход. Замыкание относительно свойства рефлексивности.

<u>Шаг 1.</u> Создать пустой список для хранения пар замыкания. Пустить цикл по i от 1 до N.

<u>Шаг 2.</u> Если $M_{ii} = 0$, пару (i, i) добавить в замыкание рефлексивности. ответ – замыкание относительно свойства рефлексивности.

Трудоемкость алгоритма $O(N)$

2.3.2 Алгоритм построения замыкания отношения относительно свойства симметричности

Вход. Матрица $M(\rho)$ бинарного отношения p размерностью $N \times N$.

Выход. Замыкание относительно свойства симметричности.

<u>Шаг 1.</u> Создать пустой список для хранения пар замыкания. Пустить цикл по i от 1 до N и цикл по j от 1 до N.

<u>Шаг 2.</u> Если $M_{ij} = 1$ и $M_{ji} = 1$, пару (i, j) добавить в замыкание симметричности. ответ – замыкание относительно свойства симметричности.

Трудоемкость алгоритма $O(N^2)$

2.3.3 Алгоритм построения замыкания отношения относительно свойства транзитивности

Вход. Матрица $M(\rho)$ бинарного отношения p размерностью $N \times N$.

Выход. Замыкание относительно свойства симметричности.

Шаг 1. Создать пустой список для хранения пар замыкания. Пустить цикл по l от 1 до N, цикл по k от 1 до N, цикл по i от 1 до N и цикл по j от 1 до N.

Шаг 2. Проверяем все значение M_{ik}, если значение равно 1, то проверяем все значения M_{kp}, если значение равно 1, то присваеваем значению M_{jp} 1, ответ – замыкание относительно свойства транзитивности.

Трудоемкость алгоритма $O(N^4)$

2.3.4 Алгоритм построения замыкания отношения относительно свойства эквивалентности

Вход. Матрица $M(\rho)$ бинарного отношения p размерностью $N \times N$.

Выход. Замыкание относительно свойства эквивалентности.

Шаг 1. По очереди вызвать алгоритмы построения замыкания рефлексивности, симметричности и транзитивности.

Шаг 2. Ответ – замыкание относительно свойства эквивалентности.

Трудоемкость алгоритма $O(N^4)$

3 Программная реализация рассмотренных алгоритмов

3.1 Результаты тестирования программы

На рисунке 1 можно увидеть работу, реализуемой программы, по рассмотренным алгоритмам.

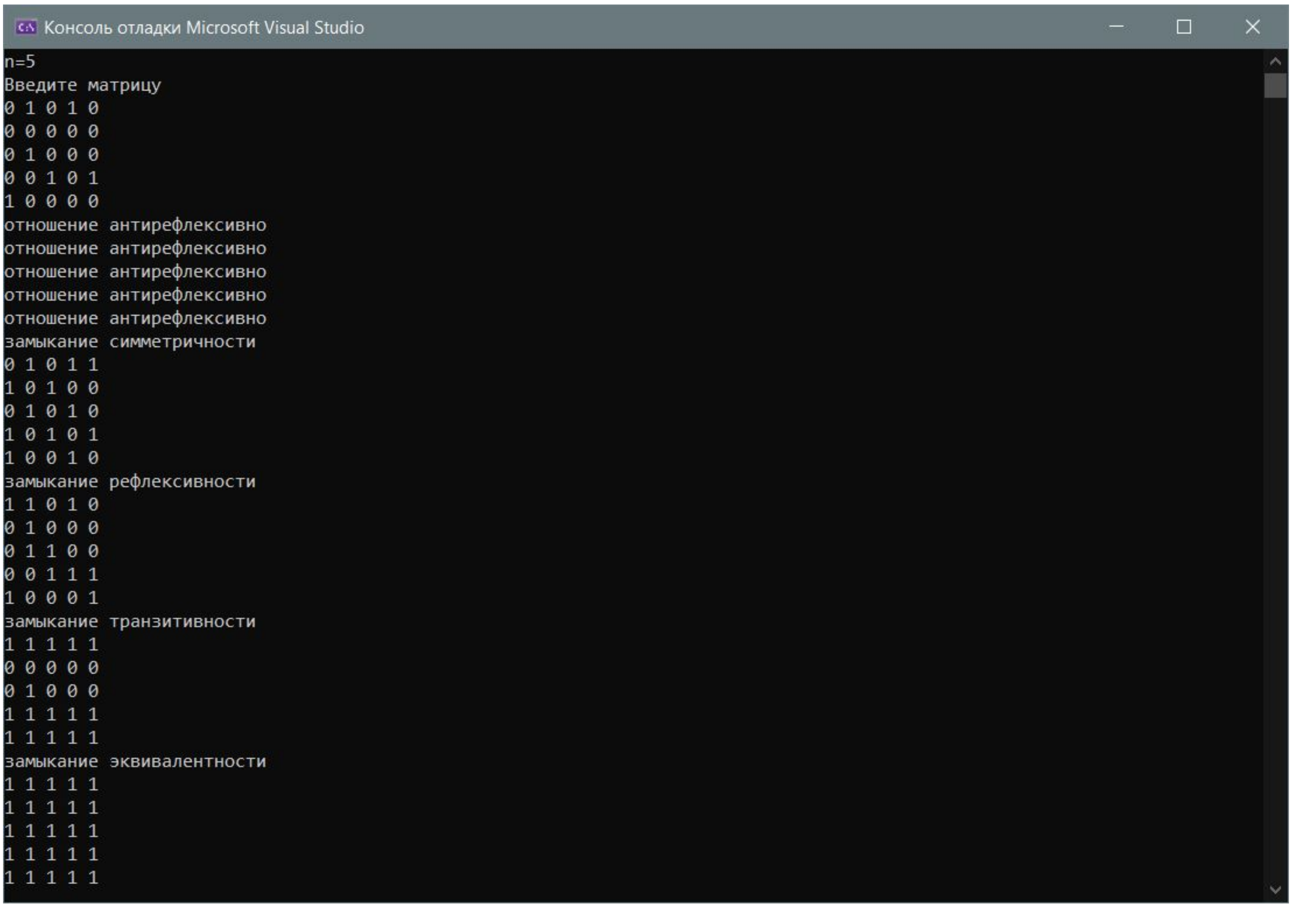

Рисунок 1 – Тест программы

3.2 Код программы, на основе рассмотренных алгоритмов, на языке C++

```
#include <iostream>

using namespace std;

int symmetry1 = 0, reflexivity1 = 0, transitivity1 = 0, antisymm

void symmetry(int** a, int n)
{
        int not_symmetry1 = 0;
```

```
        for (int i = 0; i < n; i++)
        {
                for (int j = 0; j < n; j++)
                {
                        if (a[i][j] != a[j][i])
                        not_symmetry1++;

                }
        }
        if (not_symmetry1 == 0 )
        cout << "relation is symmetrical" << endl;
        symmetry1++;
}

void antisymmetry(int** a, int n)
{
        int not_antisymmetry1 = 0;
        for (int i = 0; i < n; i++)
        {
                for (int j = 0; j < n; j++)
                {
                        if (a[i][j] == -a[j][i])
                        not_antisymmetry1++;

                }
        }
        if (not_antisymmetry1 > 0) {
                cout << "relation is antisymmetric" << endl;
                antisymmetry1++;
        }
}

void reflexivity(int** a, int n)
{
```

```
        int chek = 0;
        for (int i = 0; i < n; i++)
        {
                if (a[i][i] == 1)
                chek++;
        }
        if (chek == n) {
                cout << "relation is reflexively" << endl;
                reflexivity1++;
        }
}

void transitivity(int** a, int n)
{
        int chek = 0;
        int not_transitivity1 = 0;
        for (int i = 0; i < n; i++)
        {
                for (int j = 0; j < n; j++)
                {
                        if (a[i][j])
                        {
                        for (int k = 0; k < n; k++)
                        {
                        if (a[j][k] && !a[i][k])
                        {
                        not_transitivity1++;
                        }
                        else
                        {
                        chek++;
                        }
                        }
```

```
                    }


                }

            }
            if (chek == 1) {
                    cout << "relation is transitive" << endl;
                    transitivity1++;
            }
    }

    void pr_symmetry(int** a, int n)
    {
            for (int i = 0; i < n; i++)
            for (int j = 0; j < n; j++)
            if (a[i][j] == 1)
            a[j][i] = 1;
    }

    void pr_reflexivity(int** a, int n)
    {
            for (int i = 0; i < n; i++)
            a[i][i] = 1;
    }

    void pr_transitivity(int** a, int n)
    {
            for (int i = 0; i < n; i++)
            {
                    for (int j = 0; j < n; j++)
                    {
                            for (int k = 0; k < n; k++)
                            {
```

```
                        if (a[j][k] == 1)
                        {
                        for (int p = 0; p < n; p++)
                        {
                        if (a[k][p] == 1)
                        a[j][p] = 1;
                        }
                        }
                        }
                }
        }
}

void pr(int** a, int n, int number)
{
        int** a1;
        a1 = new int* [n];
        for (int i = 0; i < n; i++)
        {
                a1[i] = new int[n];
                for (int j = 0; j < n; j++)
                {
                        a1[i][j] = a[i][j];
                }
        }
        if (number == 1)
        {
                pr_symmetry(a1, n);
                cout << "symmetry closure" << endl;
        }
        if (number == 2)
        {
                pr_reflexivity(a1, n);
                cout << "reflexivity closure" << endl;
```

```
        }
        if (number == 3)
        {
                pr_transitivity(a1, n);
                cout << "transitivity closure" << endl;
        }
        if (number == 4)
        {
                pr_symmetry(a1, n);
                pr_reflexivity(a1, n);
                pr_transitivity(a1, n);
                cout << "equivalence closure" << endl;
        }

        for (int i = 0; i < n; i++) {
                for (int j = 0; j < n; j++)
                cout << a1[i][j] << ' ';
                cout << endl;
        }
}


int main()
{
        setlocale(LC_ALL, "RUS");
        int n;
        cout << "n=";
        cin >> n;
        int** a;
        a = new int* [n];
        cout << "Enter matrix \n";
        for (int i = 0; i < n; i++)
        {
                a[i] = new int[n];
```

```
            for (int j = 0; j < n; j++)
            {
                    cin >> a[i][j];
            }
    }

    symmetry(a, n);
    reflexivity(a, n);
    transitivity(a, n);
    antisymmetry(a, n);

    if (symmetry1 == 1 && reflexivity1 == 1 &&
                    transitivity1 == 1)
    cout << "The relation is an
                     equivalence relation" << endl;
    if (reflexivity1 == 1 && transitivity1 == 1)
    cout << "The relation is a
                    quasi-order relation" << endl;
    if (symmetry1 == 0 && reflexivity1 == 1 &&
                    transitivity1 == 1)
    cout << "The relation is a partial
                    order relation" << endl;
    pr(a, n, 1);
    pr(a, n, 2);
    pr(a, n, 3);
    pr(a, n, 4);

    return 0;
}
```

ЗАКЛЮЧЕНИЕ

В ходе выполнения лабораторной работы были рассмотрены основные свойства бинарных отношений, виды бинарных отношений, при определённых комбинациях свойств, а такжке изучена система замыканий на множестве бинарных отношений. Также были разработаны алгоритмы определения свойств отношений и их классификации.