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

Норикова Павла Сергеевича

|  |  |  |
| --- | --- | --- |
| Преподаватель  профессор, д.ф.-м.н. | \_\_\_\_\_\_\_\_\_\_ | В. А. Молчанов |
|  | подпись, дата |  |

Саратов 2022

ВВЕДЕНИЕ

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

ВИДЫ И КЛАССИФИКАЦИЯ БИНАРНЫХ ОТНОШЕНИЙ

Бинарное отношение ρ - отношение между двумя множествами A и B, то есть всякое подмножество декартова произведения этих множеств.

Бинарное отношение *ρ* на множестве *M* называется рефлексивным, если для любого элемента *a* из *M*, выполняется условие *a* *ρ* *a*.

∀ *a*∈*M*  *a* *ρ* *a* или ∀ *a*∈*M*  (*a*,*a*)∈*ρ*

Бинарное отношение *ρ* на множестве *M* называется симметричным, если для любых двух элементов *a*,*b* из *M*, из условия *a* *ρ* *b* следует условие *b* *ρ* *a*

∀*a*,*b*∈*M*  *a* *ρ* *b* → *b* *ρ* *a* или

∀ *a*,*b*∈*M*  (*a*,*b*)∈*ρ* → (*b*,*a*)∈*ρ*

Бинарное отношение *ρ* на множестве *M* называется антисимметричным, если для любых элементов *a*,*b* из *M*, из условий *a* *ρ* *b* и *b* *ρ* *a* следует условие *a*=*b*

∀ *a*,*b*,*c*∈*M*  *a* *ρ* *b* ∧ *b* *ρ* *a* → *a*=*b* или

∀ *a*,*b*∈*M*  (*a*,*b*)∈*ρ* ∧ (*b*,*a*)∈*ρ* → *a*=*b*

Бинарное отношение *ρ* на множестве *M* называется транзитивным, если для любых элементов *a*,*b*,*c* из *M*, из условий *a* *ρ* *b* и *b* *ρ* *c* следует условие *a* *ρ* *c*

∀ *a*,*b*,*c*∈*M*  *a* *ρ* *b* ∧ *b* *ρ* *c* → *a* *ρ* *c* или

∀ *a*,*b*,*c*∈*M*  (*a*,*b*)∈*ρ* ∧ (*b*,*c*)∈*ρ* → (*a*,*c*)∈*ρ*

Алгоритм 1 - проверка отношения на рефлексивность:

*Вход.* Матрица бинарного отношения v, количество элементов множества N.

*Выход.* «Отношение рефлексивно» или ничего.

Шаг 1. Заводим переменную равную нулю. В цикле по i от 0 до N - 1 проверяем значения v[i][i]. Если это значение оказалось равным нулю, прибавляем единицу к значению нашей переменной.

Шаг 2. Если значение нашей переменной равно нулю, выводим «Отношение рефлексивно».

Алгоритм 2 - проверка отношения на антирефлексивность:

*Вход.* Матрица бинарного отношения v, количество элементов множества N.

*Выход.* «Отношение антирефлексивно» или ничего.

Шаг 1. Заводим переменную равную нулю. В цикле по i от 0 до N - 1 проверяем значения v[i][i]. Если это значение оказалось равным единице, прибавляем единицу к значению нашей переменной.

Шаг 2. Если значение нашей переменной равно нулю, выводим «Отношение антирефлексивно».

Алгоритм 3 - проверка отношения на симметричность:

*Вход.* Матрица бинарного отношения v, количество элементов множества N.

*Выход.* «Отношение симметрично» или ничего.

Шаг 1. Заводим переменную равную нулю. Сравниваем все значения v[i][j] со значениями v[j][i], i,j от 0 до N - 1. Если значения оказались не равны, прибавляем единицу к значению нашей переменной.

Шаг 2. Если значение нашей переменной равно нулю, выводим «Отношение симметрично».

Алгоритм 4 - проверка отношения на антисимметричность:

*Вход.* Матрица бинарного отношения v, количество элементов множества N.

*Выход.* «Отношение антисимметрично» или ничего.

Шаг 1. Заводим переменную равную нулю. Сравниваем все значения v[i][j] со значениями v[j][i], i,j от 0 до N - 1. Если оба значения оказались равными единице, проверяем равенство i и j. Если равенства нет, прибавляем единицу к значению нашей переменной.

Шаг 2. Если значение нашей переменной равно нулю, выводим «Отношение антисимметрично».

Алгоритм 5 - проверка отношения на транзитивность:

*Вход.* Матрица бинарного отношения v, количество элементов множества N.

*Выход.* «Отношение транзитивно» или ничего.

Шаг 1. Заводим переменную равную нулю. Проверяем все значения v[i][j], i,j от 0 до N - 1. Если значение оказалось равным единице, то для нашего j проверяем все значения v[j][k], k от 0 до N - 1. Если для какого-либо k значение v[j][k] равно единице, то проверяем значение v[i][k]. Если оно равно нулю, прибавляем единицу к значению нашей переменной.

Шаг 2. Если значение нашей переменной равно нулю, выводим «Отношение транзитивно».

Бинарное отношение ρ на множестве M называется отношением эквивалентности, если оно рефлексивно, симметрично и транзитивно.

Алгоритм 6 - проверка отношения на эквивалентность:

*Вход.* Матрица бинарного отношения v, количество элементов множества N.

*Выход.* «Отношение является отношением эквивалентности» или ничего.

Шаг 1. С помощью алгоритмов 1, 3 и 5 проверим отношение на рефлексивность, симметричность и транзитивность, сохраняя при этом результаты этих проверок.

Шаг 2. Если все три результата проверки отношения положительны, выводим «Отношение является отношением эквивалентности».

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

Алгоритм 7 - проверка отношения на частичный порядок:

*Вход.* Матрица бинарного отношения v, количество элементов множества N.

*Выход.* «Отношение является отношением частичного порядка» или ничего.

Шаг 1. С помощью алгоритмов 1, 4 и 5 проверим отношение на рефлексивность, антисимметричность и транзитивность, сохраняя при этом результаты этих проверок.

Шаг 2. Если все три результата проверки отношения положительны, выводим «Отношение является отношением частичного порядка».

Бинарное отношение *ρ* на множестве А называют отношением квазипорядка, если оно рефлексивно и транзитивно.

Алгоритм 8 - проверка отношения на квазипорядок:

*Вход.* Матрица бинарного отношения v, количество элементов множества N.

*Выход.* «Отношение является отношением квазипорядка» или ничего.

Шаг 1. С помощью алгоритмов 1 и 5 проверим отношение на рефлексивность и транзитивность, сохраняя при этом результаты этих проверок.

Шаг 2. Если оба результата проверки отношения положительны, выводим «Отношение является отношением квазипорядка».

ЗАМЫКАНИЯ

Множество Z подмножеств множества A называется системой замыканий, если оно замкнуто относительно пересечений, т.е. выполняется ∩B∈Z для любого подмножества B⊂Z. В частности, для ∅⊂Z выполняется ∩∅=A∈Z.

Лемма о системах замыканий бинарных отношений.

На множестве P(A2) всех бинарных отношений между элементами множества A следующие множества являются системами замыканий:

1) Zr – множество всех рефлексивных бинарных отношений между элементами множества A,

2) Zs – множество всех симметричных бинарных отношений между элементами множества A,

3) Zt – множество всех транзитивных бинарных отношений между элементами множества A,

4) Zeq = Eq(A) – множество всех отношений эквивалентности на множестве A.

Множество Zas всех антисимметричных бинарных отношений между элементами множества A не является системой замыкания.

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

1) X ⊂ Y → f(X) ⊂ f(Y);

2) X ⊂ f(X);

3) ff(X) = f(X), для всех X,Y ∈ P(A).

Лемма о замыканиях бинарных отношений.

На множестве P(A2) всех бинарных отношений между элементами множества A

следующие отображения являются операторами замыканий:

1) fr(ρ) = ρ ∪ ΔA – наименьшее рефлексивное бинарное отношение, содержащее отношение ρ ⊂ A2,

2) fs(ρ) = ρ ∪ ρ-1 – наименьшее симметричное бинарноеотношение, содержащее отношение ρ ⊂ A2,

3) ft(ρ) = ⋃n=1 ρn – наименьшее транзитивное бинарноеотношение, содержащее отношение ρ ⊂ A2,

4) feq(ρ) = ftfsfr(ρ) – наименьшее отношение эквивалентности, содержащее отношение ρ ⊂ A2.

Алгоритм 9 – построение замыкания рефлексивности:

*Вход.* Матрица бинарного отношения v, количество элементов множества N.

*Выход.* Матрица замыкания рефлексивности v2.

Шаг 1. Создаём матрицу v2. Переносим в неё все элементы матрицы v. В цикле по i от 0 до N - 1 присваиваем значениям v2[i][i] единицу.

Шаг 2. Выводим матрицу v2.

Алгоритм 10 - построение замыкания симметричности:

*Вход.* Матрица бинарного отношения v, количество элементов множества N.

*Выход.* Матрица замыкания симметричности v2.

Шаг 1. Создаём матрицу v2. Переносим в неё все элементы матрицы v. Проверяем все значения v2[i][j], i,j от 0 до N - 1. Если значение v2[i][j] равны единице, присваиваем значениям v2[j][i] единицу.

Шаг 2. Выводим матрицу v2.

Алгоритм 11 - построение замыкания транзитивности:

*Вход.* Матрица бинарного отношения v, количество элементов множества N.

*Выход.* Матрица замыкания транзитивности v2.

Шаг 1. Создаём матрицу v2. Переносим в неё все элементы матрицы v. Проверяем все значения v2[i][j], i,j от 0 до N - 1. Если значение оказалось равным единице, то для нашего j проверяем все значения v2[j][k], k от 0 до N - 1. Если для какого-либо k значение v2[j][k] равно единице, то присваиваем значению v2[i][k] единицу.

Шаг 2. Повторяем первый шаг N раз.

Шаг 3. Выводим матрицу v2.

Алгоритм 12 построение замыкания эквивалентности:

*Вход.* Матрица бинарного отношения v, количество элементов множества N.

*Выход.* Матрица замыкания эквивалентности v2.

Шаг 1. Создаём матрицу v2. Переносим в неё все элементы матрицы v. Для матрицы v2 применяем последовательно алгоритмы 9, 10, 11.

Шаг 2. Выводим матрицу v2.

ВРЕМЕННАЯ СЛОЖНОСТЬ АЛГОРИТМОВ

Проверка на рефлексивность и антирефлексивность – O(n)

Проверка на симметричность и антисимметричность – O(n2)

Проверка на транзитивность - O(n3)

Проверка на эквивалентность - O(n3)

Проверка на частичный порядок - O(n3)

Проверка на квазипорядок - O(n3)

Построение замыкания рефлексивности - O(n)

Построение замыкания симметричности - O(n2)

Построение замыкания транзитивности - O(n4)

Построение замыкания эквивалентности - O(n4)

РЕЗУЛЬТАТЫ ТЕСТИРОВАНИЯ ПРОГРАММЫ

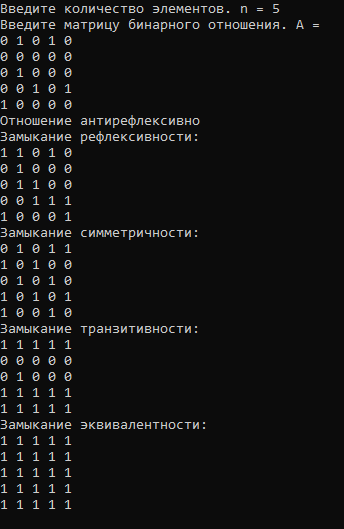


Рис.1 – первый тест

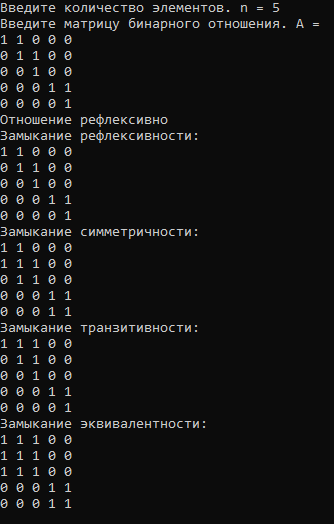


Рис.2 – второй тест

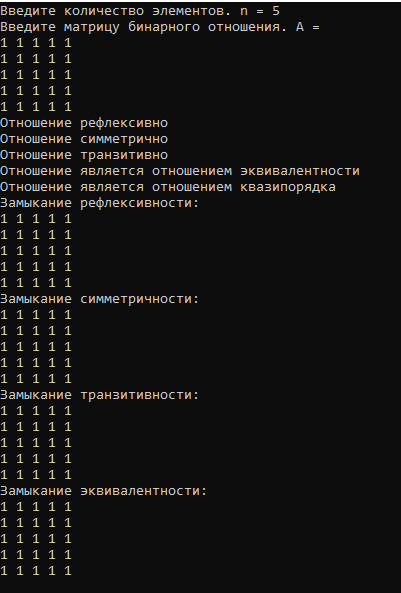


Рис.3 – третий тест

КОД ПРОГРАММЫ

#include <iostream>

using namespace std;

int r = 0, antir = 0, s = 0, antis = 0, t = 0;

void ref(int\*\* v, int n)

{

int checker = 0;

for (int i = 0; i < n; i++) {

if (v[i][i] == 1)

checker++;

}

if (checker == n) {

cout << "Отношение рефлексивно" << endl;

r++;

}

else if (checker == 0) {

cout << "Отношение антирефлексивно" << endl;

antir++;

}

}

void sim(int\*\* v, int n)

{

int checker\_not\_sim = 0;

for (int i = 0; i < n; i++) {

for (int j = i + 1; j < n; j++) {

if (v[i][j] != v[j][i])

checker\_not\_sim++;

}

}

if (checker\_not\_sim == 0) {

cout << "Отношение симметрично" << endl;

s++;

}

}

void antisim(int\*\* v, int n)

{

int checker\_not\_antisim = 0;

for (int i = 0; i < n; i++) {

for (int j = 0; j < n; j++) {

if (v[i][j] == v[j][i])

if (i != j)

checker\_not\_antisim++;

}

}

if (checker\_not\_antisim == 0) {

cout << "Отношение антисимметрично" << endl;

antis++;

}

}

void tran(int\*\* v, int n)

{

int checker = 0;

for (int i = 0; i < n; i++) {

for (int j = 0; j < n; j++) {

if (v[i][j] == 1) {

for (int k = 0; k < n; k++) {

if (v[j][k] == 1)

if (v[i][k] == 0)

checker++;

}

}

}

}

if (checker == 0) {

cout << "Отношение транзитивно" << endl;

t++;

}

}

void make\_ref(int\*\* v, int n)

{

for (int i = 0; i < n; i++)

v[i][i] = 1;

}

void make\_sim(int\*\* v, int n)

{

for (int i = 0; i < n; i++)

for (int j = 0; j < n; j++)

if (v[i][j] == 1)

v[j][i] = 1;

}

void make\_tran(int\*\* v, int n)

{

for (int i = 0; i < n; i++) {

for (int j = 0; j < n; j++) {

for (int k = 0; k < n; k++) {

if (v[j][k] == 1) {

for (int p = 0; p < n; p++) {

if (v[k][p] == 1)

v[j][p] = 1;

}

}

}

}

}

}

void print\_ar(int\*\* v, int n, int which) {

int\*\* v2;

v2 = new int\* [n];

for (int i = 0; i < n; i++) {

v2[i] = new int[n];

for (int j = 0; j < n; j++) {

v2[i][j] = v[i][j];

}

}

if (which == 1) {

make\_ref(v2, n);

cout << "Замыкание рефлексивности:" << endl;

}

if (which == 2) {

make\_sim(v2, n);

cout << "Замыкание симметричности:" << endl;

}

if (which == 3) {

make\_tran(v2, n);

cout << "Замыкание транзитивности:" << endl;

}

if (which == 4) {

make\_ref(v2, n);

make\_sim(v2, n);

make\_tran(v2, n);

cout << "Замыкание эквивалентности:" << endl;

}

for (int i = 0; i < n; i++) {

for (int j = 0; j < n; j++)

cout << v2[i][j] << ' ';

cout << endl;

}

}

int main()

{

setlocale(LC\_ALL, "Russian");

int n;

cout << "Введите количество элементов. n = ";

cin >> n;

int\*\* v;

v = new int\* [n];

cout << "Введите матрицу бинарного отношения. А = " << endl;

for (int i = 0; i < n; i++) {

v[i] = new int[n];

for (int j = 0; j < n; j++) {

cin >> v[i][j];

}

}

ref(v, n);

sim(v, n);

tran(v, n);

antisim(v, n);

if (r == 1 && s == 1 && t == 1)

cout << "Отношение является отношением эквивалентности" << endl;

if (r == 1 && antis == 1 && t == 1)

cout << "Отношение является отношением частичного порядка" << endl;

if (r == 1 && t == 1)

cout << "Отношение является отношением квазипорядка" << endl;

print\_ar(v, n, 1);

print\_ar(v, n, 2);

print\_ar(v, n, 3);

print\_ar(v, n, 4);

return 0;

}

ЗАКЛЮЧЕНИЕ

В ходе данной работы были изучены и реализованы алгоритмы проверки бинарных отношений на рефлексивность, антирефлексивность, симметричность, антисимметричность, транзитивность, эквивалентность, частичный порядок и квазипорядок, а также алгоритмы построения основных замыканий.