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

Яхина Шамиля Илдусовича

Преподаватель

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

подпись, дата

Саратов 2022

СОДЕРЖАНИЕ

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

1 Бинарные отношения

1.1 Определение бинарного отношения

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

бинарными

Для бинарного отношения $\rho \subset A \times B$ область определения D_ρ и множество значений E_ρ определяются как подмножества соответствующих множеств и по следующим формулам:

$D_\rho = \{a : (a, b) \in \rho$ для некоторого $b \in B\}$,

$E_\rho = \{b : (a, b) \in \rho$ для некоторого $a \in A\}$.

1.2 Классификация бинарных отношений

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

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

Основываясь на этом, можно выделить три типа отношений:

1. Отношение эквивалентности

   Бинарное отношение ε на множестве A называется
   отношением эквивалентности (или просто *эквивалентностью*),
   если оно рефлексивно, симметрично и транзитивно,
2. Отношение квазипорядка

   Бинарное отношение ω на множестве A называется
   отношением квазипорядка (или просто *квазипорядком*),
   если оно рефлексивно и транзитивно,
3. Отношение порядка

   Бинарное отношение ω на множестве A называется
   отношением порядка (или просто *порядком*),
   если оно рефлексивно, антисимметрично и транзитивно,

Для того, чтобы реализовать алгоритм классификации бинарных отношений, удобно пользоваться матрицей бинарного отношения.

Матрицей бинарного отношения ρ между элементами множеств $A =$

$\{a_1, ..., a_m\}$ и $B = \{b_1, ..., b_n\}$ называется прямоугольная таблица $M(\rho)$, состоящая из m строк и n столбцов, в которой на пересечении i-ой строки и j-го столбца стоит элемент $[M(\rho)]_{ij}$ из множества 0,1, определяемый по правилу:

$$[M(\rho)]_{ij} = \begin{cases} 1 & \text{, если } (a_i, b_j) \in \rho \\ 0 & \text{, в противном случае} \end{cases}$$

2 Системы замыканий

2.1 Определение системы замыканий

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

$\cap B \in Z$ для любого подмножества $B \subset Z$

2.2 Лемма 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 множетсва A не является системой замыкания.

3 Результаты работы

3.1 Описание алгоритма классификации бинарных отношений

Из пункта 1.2 следует, что в нашем алгоритме будут определяться 5 свойств бинарных отношений, а именно:

1. рефлексивность,
2. антирефлексивность,
3. симметричность,
4. антисимметричность,
5. транзитивность.

Как по матрице представления определить свойства бинарного отношения:

1. Для того, чтобы бинарное отношение было *рефлексивным*, на главной диагонали должны стоять только единицы,
2. Для того, чтобы бинарное отношение было *антирефлексивным*, на главной диагонали должны стоять только нули,
3. Для того, чтобы бинарное отношение было *симметричным*, матрица представления должна равняться транспонированной матрице,
4. Для того, чтобы бинарное отношение было *антисимметричным*, в матрице должны отсутствовать единицы, симметричные относительно главной диагонали,
5. Для того, чтобы бинарное отношение было *транзитивным*, матрица, полученная перемножением матрицы саму на себя, должна являться частью исходной матрицы бинарного отношения.

После проверки бинарного отношения на эти 5 свойств, мы можем судить, к какому типу отношений относится данное бинарное отношение. В этом и заключается алгоритм классификации.

1. Бинарное отношение является отношением эквивалентности, если выполнились следующие три свойства: рефлексивность, симметричность и транзитивность.
2. Если выполнились свойства рефлексивности и транзитивности, то это отношение является отношением квазипорядка,
3. Бинарное отношение является отношением порядка, если оно рефлексивно, антисимметрично и транзитивно.

3.2 Описание алгоритма построения основных замыканий бинарных отношений

1. Матрица *рефлексивного* замыкания равна $R \cup E_n$, т.е. необходимо все элементы главной диагонали заменить нулями,
2. Матрица *симметричного* замыкания равна $R \cup R^T$, т.е. если элемент матрицы равен единице, то симметричный ему элемент относительно главной диагонали тоже должен быть равен единице,
3. Стратегия построения матрицы *транзитивного* замыкания такова: если имеется две пары элементов (i, j) и (j, k), то необходимо добавить пару (i, k). После этого запустить цикл еще раз, т.к. после добавления новой пары этому условию могут удовлетворять еще две пары.

3.3 Код программы

```
#include <iostream>

using namespace std;
int brk = 0;

bool bo_is_reflexive(int N, int **a)
{
int res = 0;

for (int i = 0; i < N; ++i)
{
if (a[i][i] == 1)
res = 1;
else res = 0;

if (res == 0)
{
return  res;
}
}
```

```
return  res;
}

bool bo_is_antireflexive(int N, int **a)
{
int res = 0;

for (int i = 0; i < N; ++i)
{
if (a[i][i] == 0)
res = 1;
else res = 0;

if (res == 0)
{
return  res;
}
}

return  res;
}

bool bo_is_symmetric(int N, int **a)
{
int res = 0;

for (int i = 0; i < N; ++i)
{
for (int j = i + 1; j < N; ++j)
{
if (a[i][j] == a[j][i])
res = 1;
else res = 0;
```

```
if (res == 0)
{
return  res;
}
}
}

return  res;
}

bool bo_is_antisymmetric(int N, int** a)
{
int res = 0;

for (int i = 0; i < N; ++i)
{
for (int j = i + 1; j < N; ++j)
{
if (a[i][j] == 1 && a[j][i] == 1) {
if (i = j)
res = 1;
else res = 0;
}
else res = 0;

if (res == 0)
{
return  res;
}
}
}

return  res;
}
```

```
bool bo_is_transitive(int N, int** a)
{
int res = 0;

for (int i = 0; i < N; ++i)
{
for (int j = 0; j < N; ++j)
{
for (int k = 0; k < N; ++k)
{
if (a[i][j] >= a[i][k] * a[k][j])
res = 1;
else res = 0;

if (res == 0)
{
return  res;
}
}
}
}

return  res;
}

//строим замыкания
void z_reflexive(int N, int** a)
{
for (int i = 0; i < N; i++)
{
a[i][i] = 1;
}
}
```

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

void z_tranz(int N, int** a)
{
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 d = 0; d < N; d++)
if (a[k][d] == 1)
a[j][d] = 1;

}


void z_build(int N, int** a, int vvod)
{
int** z_a;
z_a = new int* [N];
for (int i = 0; i < N; i++) {
z_a[i] = new int[N];
for (int j = 0; j < N; j++) {
z_a[i][j] = a[i][j];
```

```
}
}
switch (vvod)
{
case 1:
z_reflexive(N, z_a);
break;
case 2:
z_sim(N, z_a);
break;
case 3:
z_tranz(N, z_a);
break;
case 4:
z_reflexive(N, z_a);
z_sim(N, z_a);
z_tranz(N, z_a);
break;
case 5:
brk = 1;
break;
default:
cout << "Error" << endl;
break;
}
if (brk == 0) {
cout << "Построенное замыкание:" << endl;
for (int i = 0; i < N; i++) {
for (int j = 0; j < N; j++)
cout << z_a[i][j] << ' ';
cout << endl;
}
}
}
```

```
void    bo_result(int N, int **a)
{
cout << "Введеная матрица:" << endl;
for (int i = 0; i < N; i++) {
for (int j = 0; j < N; j++)
cout << a[i][j] << ' ';
cout << endl;
}
int res_refl = bo_is_reflexive(N, a);
int res_antirefl = bo_is_antireflexive(N, a);
int res_simm = bo_is_symmetric(N, a);
int res_antisimm = bo_is_antisymmetric(N, a);
int res_tranz = bo_is_transitive(N, a);
cout << "Результаты (1 - да, 0 - нет):" << endl;
cout << "Рефлексивность:" << res_refl << endl;
cout << "Антирефлексивность:" << res_antirefl << endl;
cout << "Симметричность:" << res_simm << endl;
cout << "Антисимметричность:" << res_antisimm << endl;
cout << "Транзитивность:" << res_tranz << endl;

if (res_refl == 1 && res_antirefl == 0 &&
res_antisimm == 0 && res_tranz == 1) {
cout << "Данное отношение является отношением квазипорядка" << endl;
if (res_simm == 1)
cout << "Данное отношение является отношением эквивалентности" << endl;
}
if (res_refl == 1 && res_antirefl == 0 && res_simm == 0 &&
res_antisimm == 1 && res_tranz == 1) {
cout << "Данное отношение является отношением порядка" << endl;
}

int vvod;
cout << "Введите, какое замыкание требуется построить:" << endl;
```

```
cout << "1 - рефлексивное" << endl << "2 - симметричное" << endl <<
"3 - транзитивное" << endl << "4 - эквивалентное" << endl <<
"5 - не строить никакое замыкание" << endl;
while (brk == 0) {
cout << "Введите номер:" << endl;
cin >> vvod;
z_build(N, a, vvod);
}
}
int main()
{
setlocale(LC_ALL, "Rus");

int sposob, i, j, N;
cout << "Введите способ ввода (1 - поэлементно, 2 - построчно): ";
cin >> sposob;
cout << "Введите размерность матрицы бинарного отношения: ";
cin >> N;
int ** a;
a = new int* [N];
cout << "Введите матрицу A" << endl;
if (sposob == 1) {
for (i = 0; i < N; i++) {
a[i] = new int[N];
for (j = 0; j < N; j++) {
cout << "A["
<< i
<< "]["
<< j
<< "] = ";
cin >> a[i][j];
}
}
}
```

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

cout << endl;
bo_result(N, a);
cout << endl;
}
```

3.4 Результаты тестирования программ

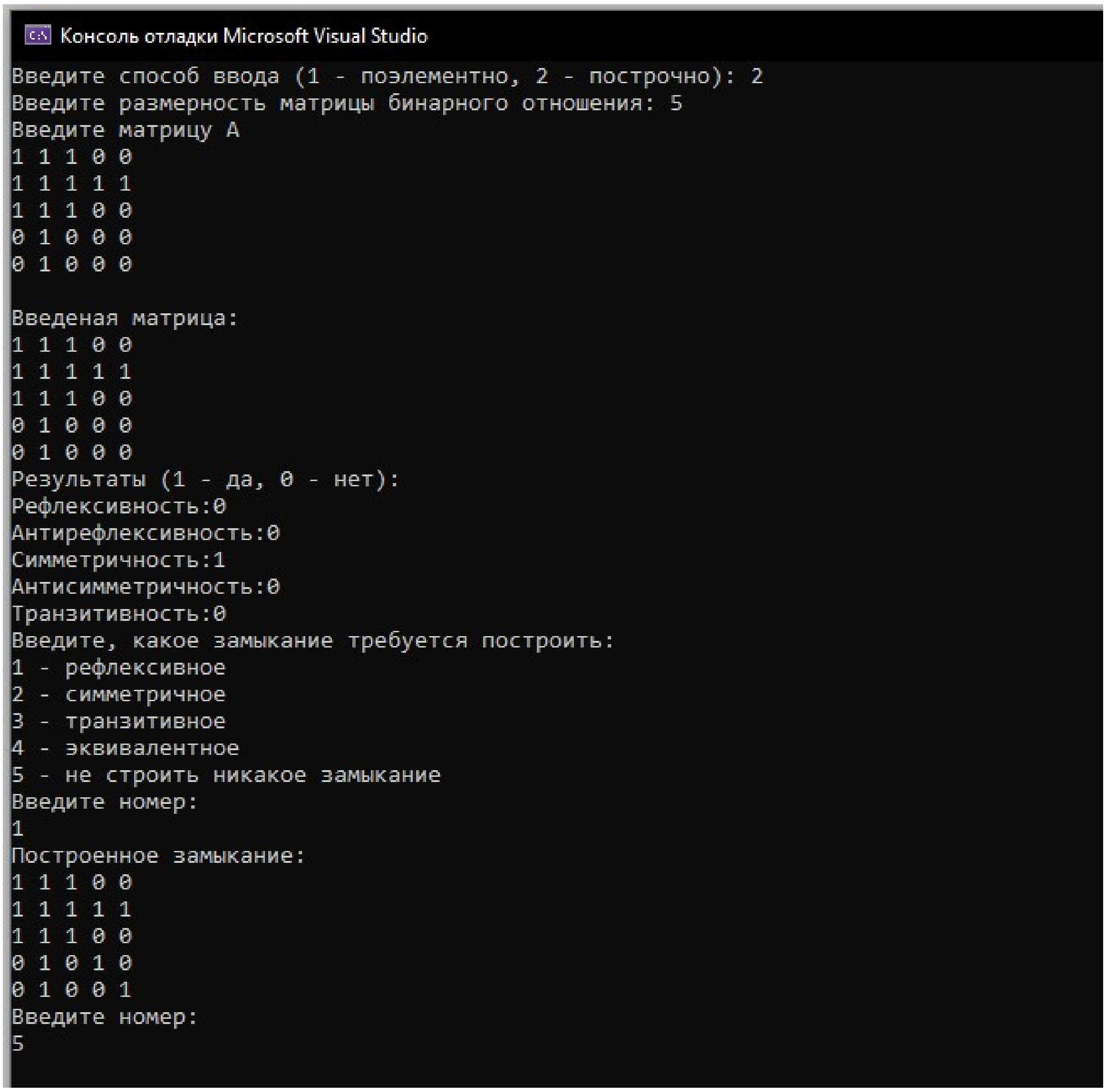
Тестирование №1:

На вход поступает матрица:

$$\begin{pmatrix} 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \end{pmatrix}$$

Она обладает свойством симметричности.

Построим рефлексивное замыкание.



```
Консоль отладки Microsoft Visual Studio
Введите способ ввода (1 - поэлементно, 2 - построчно): 2
Введите размерность матрицы бинарного отношения: 5
Введите матрицу А
1 1 1 0 0
1 1 1 1 1
1 1 1 0 0
0 1 0 0 0
0 1 0 0 0

Введеная матрица:
1 1 1 0 0
1 1 1 1 1
1 1 1 0 0
0 1 0 0 0
0 1 0 0 0
Результаты (1 - да, 0 - нет):
Рефлексивность:0
Антирефлексивность:0
Симметричность:1
Антисимметричность:0
Транзитивность:0
Введите, какое замыкание требуется построить:
1 - рефлексивное
2 - симметричное
3 - транзитивное
4 - эквивалентное
5 - не строить никакое замыкание
Введите номер:
1
Построенное замыкание:
1 1 1 0 0
1 1 1 1 1
1 1 1 0 0
0 1 0 1 0
0 1 0 0 1
Введите номер:
5
```

Рисунок 1 – Тестировние №1

Тестирование №2:

На вход поступает матрица:

$$\begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 \end{pmatrix}$$

Она обладает свойством антирефлексивности и транзитивности.

Построим все типы замыканий.

```
C:\Users\Shamil_\source\repos\ConsoleApplication41\Debug\ConsoleApplication41.exe
Введите способ ввода (1 - поэлементно, 2 - построчно): 2
Введите размерность матрицы бинарного отношения: 5
Введите матрицу А
0 0 0 0 0
1 0 0 0 0
1 1 0 0 0
1 1 0 0 0
1 1 1 0 0

Введеная матрица:
0 0 0 0 0
1 0 0 0 0
1 1 0 0 0
1 1 0 0 0
1 1 1 0 0
Результаты (1 - да, 0 - нет):
Рефлексивность:0
Антирефлексивность:1
Симметричность:0
Антисимметричность:0
Транзитивность:1
Введите, какое замыкание требуется построить:
1 - рефлексивное
2 - симметричное
3 - транзитивное
4 - эквивалентное
5 - не строить никакое замыкание
Введите номер:
1
Построенное замыкание:
1 0 0 0 0
1 1 0 0 0
1 1 1 0 0
1 1 0 1 0
1 1 1 0 1
Введите номер:
2
Построенное замыкание:
0 1 1 1 1
1 0 1 1 1
1 1 0 0 1
1 1 0 0 0
1 1 1 0 0
Введите номер:
3
Построенное замыкание:
0 0 0 0 0
1 0 0 0 0
1 1 0 0 0
1 1 0 0 0
1 1 1 0 0
Введите номер:
4
Построенное замыкание:
1 1 1 1 1
1 1 1 1 1
1 1 1 1 1
1 1 1 1 1
1 1 1 1 1
```

Рисунок 2 – Тестировние №2

Тестирование №3:

На вход поступает матрица:

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 \end{pmatrix}$$

Она обладает свойством рефлексивности, симметричности и транзитивности, а значит является отношением квазипорядка и отношением эквивалентности.

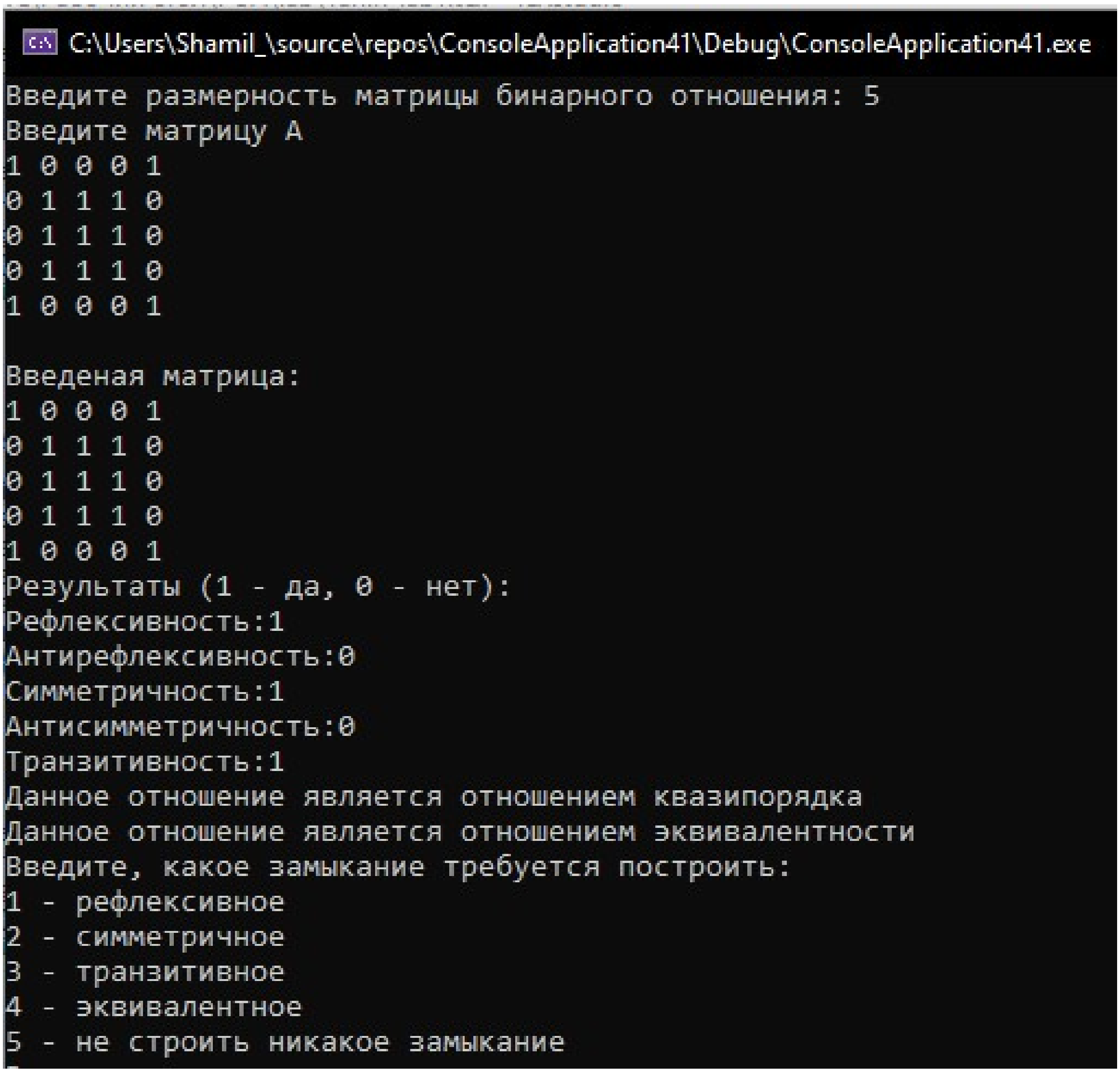


Рисунок 3 – Тестировние №3

3.5 Оценки сложности рассмотренных алгоритмов

1. Определение рефлексивности

   Временная сложность алгоритма определения рефлексивности = $O(n)$

2. Определение антирефлексивности

   Временная сложность алгоритма определения антирефлексивности = $O(n)$
3. Определение симметричности

   Временная сложность алгоритма определения симметричности = $O(n^2/2)$ = $O(n^2)$
4. Определение антисимметричности

   Временная сложность алгоритма определения антисимметричности = $O(n^2/2)$ = $O(n^2)$
5. Определение транзитивности

   Временная сложность алгоритма определения транзитивности = $O(n^3)$
6. Определение отношения эквивалентности

   Временная сложность алгоритма определения отношения эквивалентности = $O(n^3 + n^2/2 + n)$ = $O(n^3)$
7. Определение отношения квазипорядка

   Временная сложность алгоритма определения отношения эквивалентности = $O(n^3 + n^2/2 + n)$ = $O(n^3)$
8. Определение отношения порядка

   Временная сложность алгоритма определения отношения эквивалентности = $O(n^3 + n)$ = $O(n^3)$
9. Определение построения замыкания рефлексивности

   Временная сложность алгоритма определения построения замыкания рефлексивности = $O(n)$
10. Определение построения замыкания симметричности

    Временная сложность алгоритма определения построения замыкания симметричности = $O(n^2)$
11. Определение построения замыкания транзитивности

    Временная сложность алгоритма определения построения замыкания транзитивности = $O(n^4)$
12. Определение построения замыкания эквивалентности

    Временная сложность алгоритма определения построения замыкания эквивалентности = $O(n^4 + n^2 + n)$ = $O(n^4)$

ЗАКЛЮЧЕНИЕ

В данной лабораторной работе были рассмотрены и изучены следующие темы: основные определения видов бинарных отношений, свойства бинарных отношений и основные системы замыкания на множестве бинарных отношений. В третьей части работы были реализованы алгоритмы классификации бинарных отношений и построения основных замыканий бинарных отношений.