МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ

РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное бюджетное образовательное учреждение

высшего образования

«Российский государственный университет им. А.Н. Косыгина   
(Технологии. Дизайн. Искусство)»

Кафедра автоматизированных систем обработки информации и управления

Отчет по лабораторной работе №2

# по дисциплине «Теория множеств и алгоритмы на графах».

Тема: «Бинарные отношения»

Выполнил: Федурин А.В., МВА-122

Проверил: Кузьмина Т. М.

Москва 2024

**Задание**

1. Написать программу, которая демонстрирует бинарное отношение на множестве как некоторое подмножество декартово квадрата этого множества (А×А). Для этого задать некоторое множество чисел. Это множество можно задать один раз внутри программного кода, но при защите лабораторной работы, преподаватель может попросить изменить его. На экран вывести элементы декартова квадрата исходного множества, причем элементы, составляющие заданное вариантом отношение должны быть каким-то образом выделены (например, цветом).

2. Определить свойства исследуемого отношения.

Множество А – конечное множество целых чисел, a, b ϵ A.

**Индивидуальное задание**

Элемент a находится в отношении Р к элементу b, если, а является квадратом b;

**Текст программы**

namespace Algos2Lab

{

public partial class MainWindow : Window

{

private readonly List<int> \_setA = new List<int> { 1, 2, 3, 4, 9 };

public MainWindow()

{

InitializeComponent();

DisplayRelation();

DisplayProperties();

}

private void DisplayRelation()

{

var data = new ObservableCollection<RelationRow>();

foreach (var b in \_setA)

{

var row = new RelationRow

{

B = b,

A1 = $"(1, {b})",

A1Highlight = (1 == b \* b),

A2 = $"(2, {b})",

A2Highlight = (2 == b \* b),

A3 = $"(3, {b})",

A3Highlight = (3 == b \* b),

A4 = $"(4, {b})",

A4Highlight = (4 == b \* b),

A9 = $"(9, {b})",

A9Highlight = (9 == b \* b)

};

data.Add(row);

}

RelationDataGrid.ItemsSource = data;

}

private void DisplayProperties()

{

var relation = new List<Tuple<int, int>>();

foreach (var a in \_setA)

{

foreach (var b in \_setA)

{

if (a == b \* b)

{

relation.Add(Tuple.Create(a, b));

}

}

}

bool reflexive = \_setA.All(a => relation.Contains(Tuple.Create(a, a)));

reflexive = false;

bool antiReflexive = \_setA.All(a => !relation.Contains(Tuple.Create(a, a)));

antiReflexive = true;

bool symmetric = true;

foreach (var pair in relation)

{

if (!relation.Contains(Tuple.Create(pair.Item2, pair.Item1)))

{

symmetric = false;

break;

}

}

symmetric = false;

bool antiSymmetric = true;

foreach (var pair in relation)

{

if (relation.Contains(Tuple.Create(pair.Item2, pair.Item1)) && pair.Item1 != pair.Item2)

{

antiSymmetric = false;

break;

}

}

antiSymmetric = true;

bool transitive = true;

foreach (var a in \_setA)

{

foreach (var b in \_setA)

{

if (relation.Contains(Tuple.Create(a, b)))

{

foreach (var c in \_setA)

{

if (relation.Contains(Tuple.Create(b, c)) && !relation.Contains(Tuple.Create(a, c)))

{

transitive = false;

goto Exit;

}

}

}

}

}

Exit:

transitive = true;

label1.Content = $"Рефлексивное: {reflexive}\n" +

$"Антирефлексивное: {antiReflexive}\n" +

$"Симметричное: {symmetric}\n" +

$"Антисимметричное: {antiSymmetric}\n" +

$"Транзитивное: {transitive}";

}

public class RelationRow

{

public int B { get; set; }

public string A1 { get; set; }

public bool A1Highlight { get; set; }

public string A2 { get; set; }

public bool A2Highlight { get; set; }

public string A3 { get; set; }

public bool A3Highlight { get; set; }

public string A4 { get; set; }

public bool A4Highlight { get; set; }

public string A9 { get; set; }

public bool A9Highlight { get; set; }

}

}

public class BackgroundConverter : IValueConverter

{

public object Convert(object value, Type targetType, object parameter, CultureInfo culture)

{

if (value is bool isHighlighted)

{

return isHighlighted ? Brushes.LightGreen : Brushes.White;

}

return Brushes.White;

}

public object ConvertBack(object value, Type targetType, object parameter, CultureInfo culture)

{

throw new NotImplementedException();

}

}

}}

Скрин демонстрирующий работу программы представлен на рис. 1.

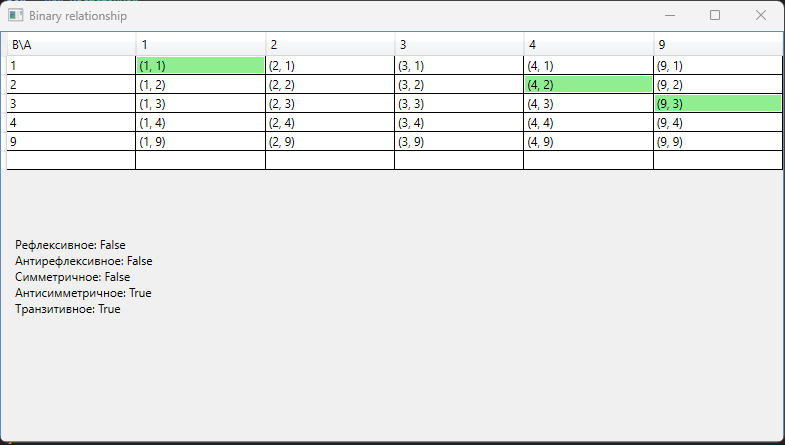


Рис. 1. Скрин работы программы.

**Описание свойств, исследуемого отношения**

Краткое описание свойств отношения P на множестве A = {1, 2, 3, 4, 9}, где a = b²:

*Рефлексивное: Ложь*

Отношение является рефлексивным, если каждый элемент связан сам с собой. Здесь присутствует только (1,1), но отсутствуют (2,2), (3,3), (4,4) и (9,9). Таким образом, отношение не является рефлексивным.

*Антирефлексивное: Ложь*

Отношение является антирефлексивным, если ни один элемент не связан с самим собой. Поскольку (1,1) присутствует, отношение не является антирефлексивным.

*Симметричное: Ложь*

Отношение симметрично, если для каждого (a, b) в P, (b, a) также находится в P. Здесь (4,2) и (9,3) находятся в P, но их инверсии (2,4) и (3,9) не находятся. Следовательно, отношение не является симметричным.

*Антисимметричное: Истинно*

Отношение антисимметрично, если всякий раз, когда оба (a, b) и (b, a) находятся в P, a должно быть равно b. Единственная пара, где оба (a, b) и (b, a) находятся в P, — это (1,1), которая удовлетворяет a = b. Таким образом, отношение антисимметрично.

*Транзитивное: Истинно*

Отношение является транзитивным, если всякий раз, когда (a, b) и (b, c) находятся в P, тогда (a, c) также находится в P. В данном случае нет таких пар, которые бы нарушали транзитивность, поэтому отношение является транзитивным.

**Выводы**

Программа демонстрирует бинарное отношение на множестве A, где отношение P определяется условием a = b². Она отображает все пары из декартова произведения A×A в DataGrid, выделяя те, которые удовлетворяют условию отношения. В настоящее время множество A имеет вид {1, 2, 3, 4, 9}, но его можно легко изменить.

Пользовательский интерфейс гарантирует, что DataGrid правильно отобразит и выделит соответствующие пары, а программа разработана таким образом, чтобы быть гибкой для различных наборов A.