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 №1

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

Тема: «Операции над множествами»

Выполнил: Винникова Е.В., МВА-122

Проверил: Кузьмина Т. М.

Москва 2024

Этапы выполнения задания №1.

В индивидуальном задании дана формула алгебры множеств (формулу нельзя преобразовывать!).

1. Написать и отладить программу, которая позволяет выполнить действия, задаваемые формулой, определенной вариантом задания. Множества можно задать один раз внутри программного кода, универсальное множество определяется как объединение всех заданных множеств. На экран должны быть выведены сами множества и результаты вычислений «по действиям». Для каждого действия определяется отдельная кнопка и отдельное поле вывода.
2. Разработать тестовый пример. Для этого:
3. Задать 4 множества натуральных чисел A, B, C, D. В формуле может присутствовать 3 множество, но задать надо все 4 множества. Значения, для задания множеств, нужно брать на интервале [10\*n,10\*n+20], где n - номер варианта. Задавать множества можно вручную, а можно с помощью генератора случайных чисел. Все множества должны попарно пересекаться, так же должны 7 пересекаться все тройки множеств. Пересечение всех 4-х множеств может быть пустым. При поверке программы преподаватель может изменить любое множество.
4. Универсальное множество определяется, как объединение всех заданных множеств.
5. Запустить программу на тестовом примере, сделать скрины, демонстрирующие работу программы по вычислению формулы.
6. В отчете для каждого скрина, на котором показан результат вычислений по определенной формуле, построить диаграмму Эйлера-Венна.
7. Обязательно указать, каким способом закрашено итоговое множество. На диаграмму нанести числа – элементы тестовых множеств. Например, пусть множество А={1,2,3,4}, В={3,4,5,6}, тогда разность А\В на рисунке 6 закрашена серым цветом.

Индивидуальное задание показано на рис. 1

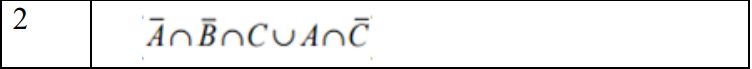


Рис. 1. Индивидуальное задание

Начальные данные множеств показаны на рис. 2

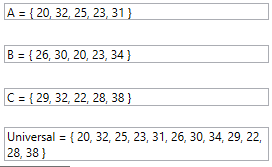


Рис. 2. Множества

Действие 1 показано на рис. 3.



Рис. 3. Действие 1

Диаграмма показана на рис. 4.

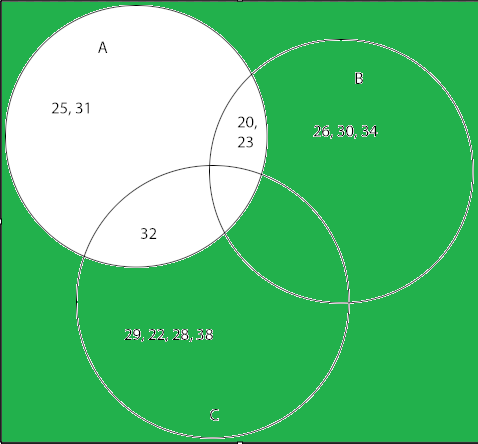


Рис. 4. Диаграмма

Действие 2 показано на рис. 5.



Рис. 5. Действие 2

Диаграмма показана на рис. 6.

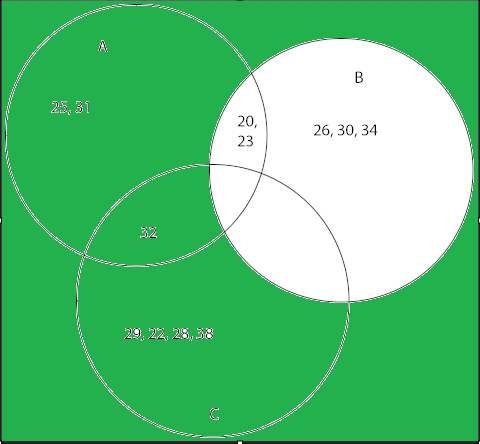


Рис. 6. Диаграмма

Действие 3 показано на рис. 7.



Рис. 7. Действие 3

Диаграмма показана на рис. 8.

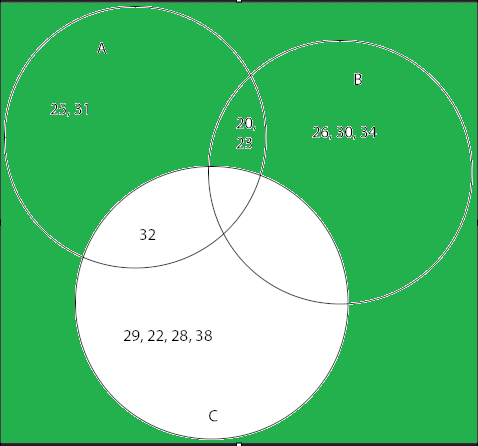


Рис. 8. Диаграмма

Действие 4 показано на рис. 9.



Рис. 9. Действие 4

Диаграмма показана на рис. 10.

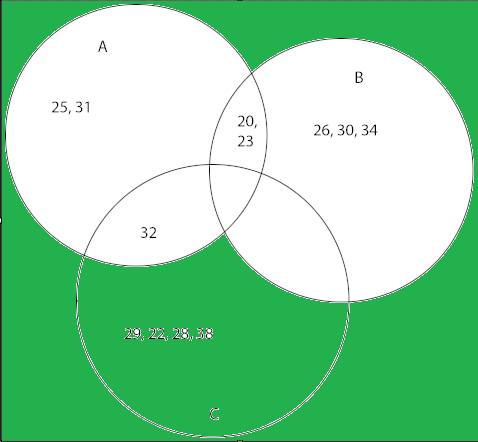


Рис. 10. Диаграмма

Действие 5 показано на рис. 11.



Рис. 11. Действие 5

Диаграмма показана на рис. 12.

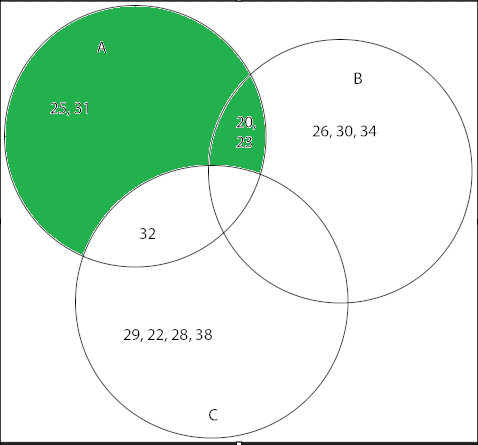


Рис. 12. Диаграмма

Действие 6 C показано на рис. 13.



Рис. 13. Действие 6

Диаграмма показана на рис. 14.

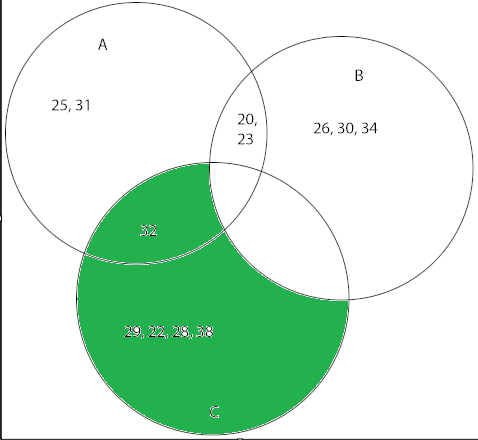


Рис. 14. Диаграмма

Действие 7 C ∪ показано на рис. 15.

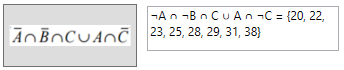


Рис. 15. Действие 7

Диаграмма показана на рис. 16.

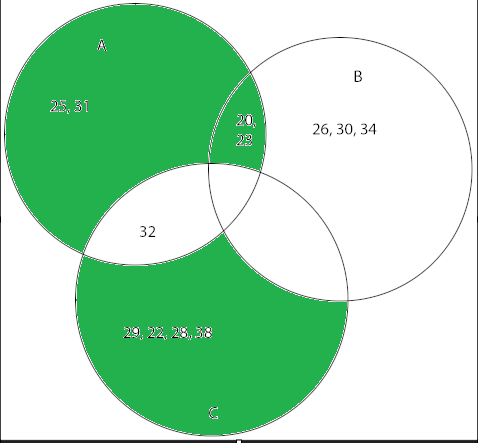


Рис. 16. Диаграмма

Код программы:

namespace Algos1Lab

{

public partial class MainWindow : Window

{

public MainWindow()

{

InitializeComponent();

InitializeSets();

DisplaySets();

}

private HashSet<int> \_setA, \_setB, \_setC, \_universalSet;

private void \_1\_Click(object sender, RoutedEventArgs e)

{

var result = GetNotA();

TextBox1.Text = $"¬A = {{{string.Join(", ", result)}}}";

}

private List<int> GetNotA()

{

var complementA = new HashSet<int>(\_universalSet);

complementA.ExceptWith(\_setA);

var sortedResult = complementA.ToList();

sortedResult.Sort();

return sortedResult;

}

private void \_2\_Click(object sender, RoutedEventArgs e)

{

var result = GetNotB();

TextBox2.Text = $"¬B = {{{string.Join(", ", result)}}}";

}

private List<int> GetNotB()

{

var complementB = new HashSet<int>(\_universalSet);

complementB.ExceptWith(\_setB);

var sortedResult = complementB.ToList();

sortedResult.Sort();

return sortedResult;

}

private void \_3\_Click(object sender, RoutedEventArgs e)

{

var result = GetNotC();

TextBox3.Text = $"¬C = {{{string.Join(", ", result)}}}";

}

private List<int> GetNotC()

{

var complementC = new HashSet<int>(\_universalSet);

complementC.ExceptWith(\_setC);

var sortedResult = complementC.ToList();

sortedResult.Sort();

return sortedResult;

}

private void \_4\_Click(object sender, RoutedEventArgs e)

{

var result = GetNotAIntersectNotB();

TextBox4.Text = $"¬A ∩ ¬B = {{{string.Join(", ", result)}}}";

}

private List<int> GetNotAIntersectNotB()

{

var notA = GetNotA();

var notB = GetNotB();

var intersection = new HashSet<int>(notA);

intersection.IntersectWith(notB);

var sortedResult = intersection.ToList();

sortedResult.Sort();

return sortedResult;

}

private void \_5\_Click(object sender, RoutedEventArgs e)

{

var result = GetAIntersectNotC();

TextBox5.Text = $"A ∩ ¬C = {{{string.Join(", ", result)}}}";

}

private List<int> GetAIntersectNotC()

{

var notC = GetNotC();

var intersection = new HashSet<int>(\_setA);

intersection.IntersectWith(notC);

var sortedResult = intersection.ToList();

sortedResult.Sort();

return sortedResult;

}

private void \_6\_Click(object sender, RoutedEventArgs e)

{

var result = GetNotBIntersectC();

TextBox6.Text = $"¬B ∩ C = {{{string.Join(", ", result)}}}";

}

private List<int> GetNotBIntersectC()

{

var notB = GetNotB();

var intersection = new HashSet<int>(notB);

intersection.IntersectWith(\_setC);

var sortedResult = intersection.ToList();

sortedResult.Sort();

return sortedResult;

}

private void \_7\_Click(object sender, RoutedEventArgs e)

{

var result = GetNotAIntersectNotBIntersectCOrAIntersectNotC();

TextBox7.Text = $"¬A ∩ ¬B ∩ C ∪ A ∩ ¬C = {{{string.Join(", ", result)}}}";

}

private List<int> GetNotAIntersectNotBIntersectCOrAIntersectNotC()

{

// ¬A ∩ ¬B ∩ C

var notA = GetNotA();

var notB = GetNotB();

var part1 = new HashSet<int>(notA);

part1.IntersectWith(notB);

part1.IntersectWith(\_setC);

// A ∩ ¬C

var notC = GetNotC();

var part2 = new HashSet<int>(\_setA);

part2.IntersectWith(notC);

// Union of both parts

var unionResult = new HashSet<int>(part1);

unionResult.UnionWith(part2);

var sortedResult = unionResult.ToList();

sortedResult.Sort();

return sortedResult;

}

private void InitializeSets()

{

\_setA = new HashSet<int> { 20, 32, 25, 23, 31 };

\_setB = new HashSet<int> { 26, 30, 20, 23, 34 };

\_setC = new HashSet<int> { 29, 32, 22, 28, 38 };

\_universalSet = new HashSet<int>(\_setA);

\_universalSet.UnionWith(\_setB);

\_universalSet.UnionWith(\_setC);

}

private void DisplaySets()

{

TextBoxSetA.Text = $"A = {{ {string.Join(", ", \_setA)} }}";

TextBoxSetB.Text = $"B = {{ {string.Join(", ", \_setB)} }}";

TextBoxSetC.Text = $"C = {{ {string.Join(", ", \_setC)} }}";

TextBoxUniversal.Text = $"Universal = {{ {string.Join(", ", \_universalSet)} }}";

}

}

}

Скриншот формы показан на рис. 17.

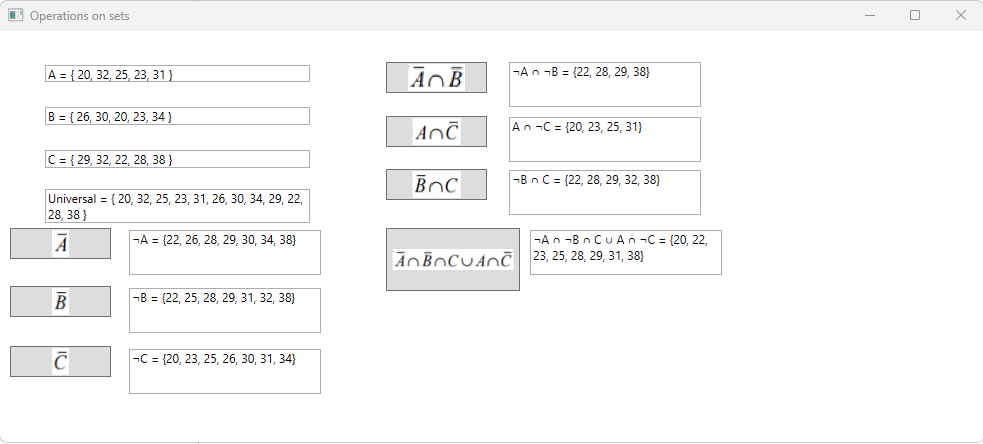


Рис. 17. Форма

Этапы выполнения задания №2.

* 1. Построить диаграмму Эйлера-Венна для множества, заданного формулой, расположенной слева от знака = (или ⊆).
  2. Построить диаграмму Эйлера-Венна для множества, заданного формулой расположенной, справа от знака = (или ⊆).
  3. Сравнить итоговые множества обеих формул и сделать выводы.

Индивидуальное задание показано на рис. 18.



Рис. 18. Индивидуальное задание

Диаграмма левой части:

Действие 1 B показано на рис. 19.

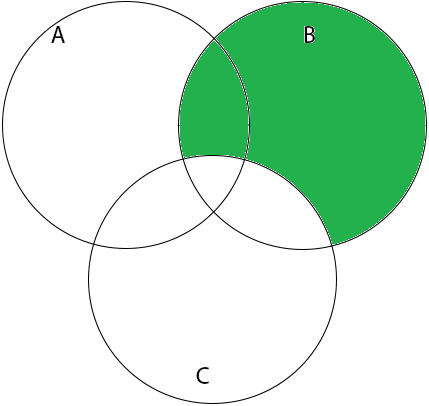


Рис. 19. Диаграмма действие 1

Действие 2 A ∪ B показано на рис. 20

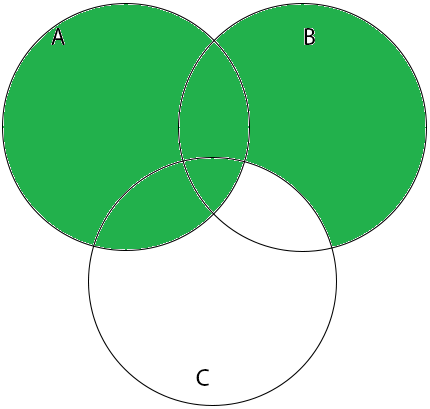


Рис. 20. Диаграмма действие 2

Действие 3 показано на рис. 21

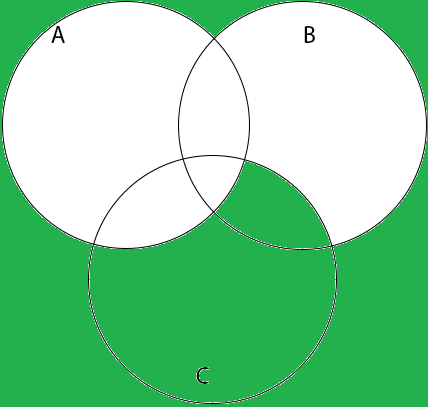


Рис. 21. Диаграмма действие 3

Диаграмма правой части:

Действие 1 C показано на рис. 22

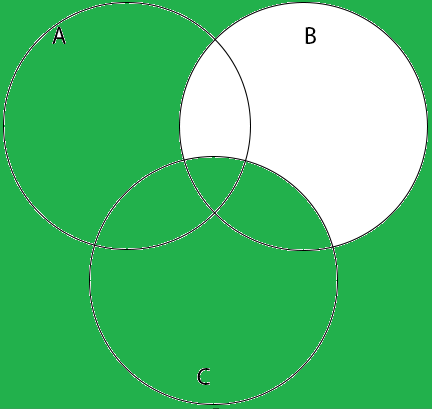


Рис. 22. Диаграмма действие 1

Действие 2 показано на рис. 23

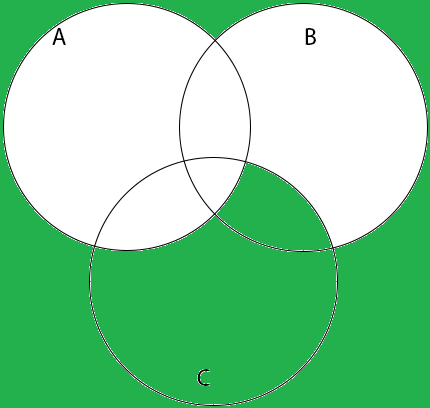


Рис. 23. Диаграмма действие 2

Вывод: Множество по 2-й формуле равно множеству из 1-й формулы.