МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ
ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Кафедра «Вычислительная техника»

ОТЧЕТ

По лабораторной работе №6

«Унарные и бинарные операции над графами»

По дисциплине «Л и ОА в ИЗ»

Выполнили: ст. гр. 22ВВ4
Жуков Илья
Чумаев Сабит

Приняли: Юрова О.В.
Акифьев И.В.

2023

Цель работы:

Написать код программы, выполнив следующие задания:

По заданию 1:

1. Сгенерируйте (используя генератор случайных чисел) две матрицы M_1, M_2 смежности неориентированных помеченных графов G_1, G_2. Выведите сгенерированные матрицы на экран.
2. * Для указанных графов преобразуйте представление матриц смежности в списки смежности. Выведите полученные списки на экран.

По заданию 2:

1. Для матричной формы представления графов выполните операцию:

а) отождествления вершин
б) стягивания ребра
в) расщепления вершины
Номера выбираемых для выполнения операции вершин ввести с клавиатуры.
Результат выполнения операции выведите на экран.
2.* Для представления графов в виде списков смежности выполните операцию:
а) отождествления вершин
б) стягивания ребра
в) расщепления вершины
Номера выбираемых для выполнения операции вершин ввести с клавиатуры.
Результат выполнения операции выведите на экран.

По заданию 3:

1. Для матричной формы представления графов выполните операцию:

а) объединения $G = G_1 \cup G_2$

б) пересечения $G = G_1 \cap G_2$

в) кольцевой суммы $G = G_1 \oplus G_2$

Результат выполнения операции выведите на экран.

По заданию 4:*

1. Для матричной формы представления графов выполните операцию декартова произведения графов $G = G_1 \times G_2$.

Результат выполнения операции выведите на экран.

Ход работы:

<u>Описание кода программы по заданию 1:</u>

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace laba6
{
    internal class Program
    {
        static void Main(string[] args)
        {
            Console.Write("Введите размер 1 матрицы: ");
            int sizeOne = Convert.ToInt32(Console.ReadLine());
            Console.Write("Введите размер 2 матрицы: ");
            int sizeTwo = Convert.ToInt32(Console.ReadLine());
            int[,] M1 = GenerateAdjacencyMatrix(sizeOne);
            int[,] M2 = GenerateAdjacencyMatrix(sizeTwo);

            Console.WriteLine("Матрица смежности для графа G1:");
            PrintMatrix(M1);

            Console.WriteLine("Матрица смежности для графа G2:");
            PrintMatrix(M2);

            List<List<int>> adjList1 = ConvertToAdjacecnyList(M1);
            List<List<int>> adjList2 = ConvertToAdjacecnyList(M2);
```

```
            Console.WriteLine("Список смежности для графа G1:");
            PrintAdjacencyList(adjList1);

            Console.WriteLine("Список смежности для графа G2:");
            PrintAdjacencyList(adjList2);
        }

        // Метод для генерации матрицы смежности
        // Метод создает объект класса Random для генерации случайных чисел и затем заполняет элементы матрицы
        // случайными значениями 0 или 1, за исключением диагональных элементов, которые остаются равными 0.
        private static int[,] GenerateAdjacencyMatrix(int size)
        {
            Random rand1 = new Random();

            int[,] matrix = new int[size, size];

            for (int i = 0; i < size; i++)
            {
                for (int j = 0; j < size; j++)
                {
                    if (i != j)
                    {
                        matrix[i, j] = rand1.Next(2);
                        matrix[j, i] = matrix[i, j];
                    }
                }
            }
            return matrix;
        }
```

```
        // Метод для вывода матрицы на экран
        static void PrintMatrix(int[,] matrix)
        {
            int size = matrix.GetLength(0);

            for (int i = 0; i < size; i++)
            {
                for (int j = 0; j < size; j++)
                {
                    Console.Write(matrix[i, j] + " ");
                }
                Console.WriteLine();
            }
            Console.WriteLine();
        }

        // Метод для преобразования матрицы смежности в список смежности
        static List<List<int>> ConvertToAdjacecnyList(int[,] matrix)
        {
            //объявление переменной size и присваивание ей значения равного длине первого измерения массива matrix.
            int size = matrix.GetLength(0);
            List<List<int>> adjList = new List<List<int>>();

            for (int i = 0; i < size; i++)
            {
                //Объявление переменной neighbors и создание нового пустого списка целых чисел для хранения смежных вершин.
                List<int> neighbors = new List<int>();
```

```
                for (int j = 0; j < size; j++)
                {
                    if (matrix[i, j] == 1)
                    {
                        neighbors.Add(j + 1);
                    }
                }
                //добавление списка neighbors в список adjList, представляющий список смежности для вершины i.
                adjList.Add(neighbors);
            }
            return adjList;
        }
        // Метод для вывода списка смежности на экран
        // Метод проходит по каждой вершине в списке и для каждой вершины выводит ее номер и список смежных вершин.
        // Каждая вершина и ее смежные вершины выводятся в формате номер_вершины: список_смежных_вершин.
        static void PrintAdjacencyList(List<List<int>>adjList)
        {
            int size = adjList.Count;

            for (int i = 0; i < size; i++)
            {
                Console.Write(i+1 + ": ");

                foreach (int neihbor in adjList[i])
                {
                    Console.Write(neihbor + " ");
                }
                Console.WriteLine();
```

```
            }

            Console.WriteLine();

        }

    }

}
```

<u>Описание кода программы по заданию 2(1 часть):</u>

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace laba6
{
    internal class Program
    {
        static void Main(string[] args)
        {
            string menuOption;
            int[] verticesToIldentify;
            int[] verticesToContract;
            int vertexToSplit;

            Console.Write("Введите размер 1 матрицы: ");
            int sizeOne = Convert.ToInt32(Console.ReadLine());
            Console.Write("Введите размер 2 матрицы: ");
            int sizeTwo = Convert.ToInt32(Console.ReadLine());

            int[,] M1 = GenerateAdjacencyMatrix(sizeOne);
            int[,] M2 = GenerateAdjacencyMatrix(sizeTwo);

            do
            {
                Console.WriteLine("Выберите опцию:");
                Console.WriteLine("1.Просмотр матрицы (1)");
                Console.WriteLine("2.Отождествление вершин (1)");
                Console.WriteLine("3.Стягивание ребра (1)");
                Console.WriteLine("4.Расщепление вершины (1)" + "\n");

                Console.WriteLine("5.Просмотр матрицы (2)");
                Console.WriteLine("6.Отождествление вершин (2)");
                Console.WriteLine("7.Стягивание ребра (2)");
                Console.WriteLine("8.Расщепление вершины (2)" + "\n");

                Console.WriteLine("9.Выход");

                menuOption = Console.ReadLine();
```

```
                switch (menuOption)
                {
                    case "1":
                        Console.WriteLine("Матрица смежности для графа
G1:");
                        PrintMatrix(M1);
                        break;
                    case "2":
                        Console.Write("Введите   номера   вершин   для
отождествления (через пробел):");
                  verticesToIldentify   =   Console.ReadLine().Split('
').Select(int.Parse).ToArray();
                  IdentifyVertices(ref   M1,   verticesToIldentify[0],
verticesToIldentify[1]);
                        break;
                    case "3":
                  Console.Write("Введите номера вершин для стягивания
ребра (через пробел): ");
                  verticesToContract    =    Console.ReadLine().Split('
').Select(int.Parse).ToArray();
                        ContractEdge(ref   M1,   verticesToContract[0],
verticesToContract[1]);
                        break;
                    case "4":
                  Console.Write("Введите номер вершины для расщепления:
");
                        vertexToSplit = int.Parse(Console.ReadLine());
                        SplitVertex(ref M1, vertexToSplit);
                        break;
                    case "5":
                        Console.WriteLine("Матрица смежности для графа
G2:");
                        PrintMatrix(M2);
                        break;
                    case "6":
                        Console.Write("Введите   номера   вершин   для
отождествления (через пробел):");
                  verticesToIldentify   =   Console.ReadLine().Split('
').Select(int.Parse).ToArray();
                  IdentifyVertices(ref   M2,   verticesToIldentify[0],
verticesToIldentify[1]);
                        break;
                    case "7":
                  Console.Write("Введите номера вершин для стягивания
ребра (через пробел): ");
                  verticesToContract    =    Console.ReadLine().Split('
').Select(int.Parse).ToArray();
                        ContractEdge(ref   M2,   verticesToContract[0],
verticesToContract[1]);
                        break;
```

```
                    case "8":
                   Console.Write("Введите номер вершины для расщепления:
");
                        vertexToSplit = int.Parse(Console.ReadLine());
                        SplitVertex(ref M2, vertexToSplit);
                        break;
                    case "9":
                        Console.WriteLine("Программа завершена");
                        break;
                    default:
                        Console.WriteLine("Неверная  опция,  попробуйте
ещё раз");
                        break;
                }
                Console.WriteLine();
            } while (menuOption != "9");
        }
        //Метод  GenerateAdjacencyMatrix  генерирует  случайную  матрицу
смежности для графа заданного размера.
        private static int[,] GenerateAdjacencyMatrix(int size)
        {
            Random rand1 = new Random();

            int[,] matrix = new int[size, size];

            for (int i = 0; i < size; i++)
            {
                for (int j = i + 1; j < size; j++)
                {
                    matrix[i, j] = rand1.Next(2);
                    matrix[j, i] = matrix[i, j];
                }
            }
            return matrix;
        }
        // Метод для вывода матрицы на экран
        static void PrintMatrix(int[,] matrix)
        {
            int size = matrix.GetLength(0);

            for (int i = 0; i < size; i++)
            {
                for (int j = 0; j < size; j++)
                {
                    Console.Write(matrix[i, j] + " ");
                }
                Console.WriteLine();
            }
            Console.WriteLine();
        }
        //Метод осуществляет отождествления вершин
```

```
      static void IdentifyVertices(ref int[,] matrix, int vertex1, int
vertex2)
        {
         //Сначала определяется размерность массива matrix с помощью
метода GetLength(0) и присваивается переменной size.
            int size = matrix.GetLength(0);

            //Затем начинается первый цикл for для итерации по строкам
массива matrix. Внутри цикла проверяется, что текущий индекс i не равен
vertex1 и vertex2.
         //Если это условие выполняется и элемент matrix[i, vertex2]
равен 1, то элемент matrix[i, vertex1] присваивается значение 1
            for (int i = 0; i < size; i++)
            {
                if (i != vertex1 && i != vertex2)
                {
                    if (matrix[i, vertex2] == 1)
                    {
                        matrix[i, vertex1] = 1;
                    }
                }
            }
            //После этого начинается второй цикл for для итерации по
столбцам массива matrix. Внутри цикла проверяется, что текущий индекс
j не равен vertex1 и vertex2.
          //Если это условие выполняется и элемент matrix[vertex2, j]
равен 1, то элемент matrix[vertex1, j] присваивается значение 1
            for (int j = 0; j < size; j++)
            {
                if (j != vertex1 && j != vertex2)
                {
                    if (matrix[vertex2, j] == 1)
                    {
                        matrix[vertex1, j] = 1;
                    }
                }
            }
            //Затем следуют два цикла for, которые устанавливают все
элементы в строке vertex2 и столбце vertex2 равными 0
            for (int i = 0; i < size; i++)
            {
                matrix[i, vertex2] = 0;
            }

            for (int j = 0; j < size; j++)
            {
                matrix[vertex2, j] = 0;
            }
        }
        //Метод осуществляет стягивание ребра
```

```
        static void ContractEdge(ref int[,] matrix, int vertex1, int
vertex2)
        {
         //Сначала определяется размерность массива matrix с помощью
метода GetLength(0) и присваивается переменной size.
            int size = matrix.GetLength(0);

         //Затем вызывается метод IdentifyVertices, который изменяет
значения элементов массива matrix в соответствии с определенными
условиями, связанными с вершинами vertex1 и vertex2.
            IdentifyVertices(ref matrix, vertex1, vertex2);

            //После этого создается новый двумерный массив newMatrix
размером (size - 1) x (size - 1).
            int[,] newMatrix = new int[size - 1, size - 1];
            int newRow = 0;
            int newCol = 0;

            //Затем начинается первый цикл for для итерации по строкам
массива matrix.
            for (int i = 0; i < size; i++)
            {
            //Внутри цикла проверяется, что текущий индекс i не равен
vertex2
                if (i != vertex2)
                {
               //Если это условие выполняется, то начинается второй
цикл for для итерации по столбцам массива matrix.
                    for (int j = 0; j < size; j++)
                    {
                  //Внутри второго цикла проверяется, что текущий
индекс j не равен vertex2.
                        if (j != vertex2)
                        {
                            //Если это условие выполняется, то элемент
matrix[i, j] присваивается элементу newMatrix[newRow, newCol], а затем
значение newCol увеличивается на 1.
                            newMatrix[newRow, newCol] = matrix[i, j];
                            newCol++;
                        }
                    }
                //После завершения второго цикла for значение newRow
увеличивается на 1, а newCol сбрасывается в 0.
                    newRow++;
                    newCol = 0;
                }
            }

            matrix = newMatrix;
        }
```

```
        //Метод осуществляет расщепление вершины
        //Ключевое слово ref указывает, что массив будет изменяться непосредственно в вызывающем коде.
        static void SplitVertex(ref int[,] matrix, int vertex)
        {
            //получение размера матрицы matrix. Метод GetLength(0) возвращает длину массива по указанному измерению (в данном случае по оси 0)
            int size = matrix.GetLength(0);

         //создание новой матрицы newMatrix с размерностью на 1 больше исходной матрицы matrix. Это нужно для добавления новой вершины.
            int[,] newMatrix = new int[size + 1, size + 1];

         // Вложенный цикл for используется для копирования существующих связей в новую матрицу:
         // В этом коде мы пробегаем по каждому элементу исходной матрицы matrix и копируем его в
         // новую матрицу newMatrix, за исключением связей с вершиной, которую мы разделяем.
            for (int i = 0; i < size; i++)
            {
                for (int j = 0; j < size; j++)
                {
                    if (i != vertex && j != vertex)
                    {
                        if (i > vertex && j > vertex)
                        {
                            newMatrix[i + 1, j + 1] = matrix[i, j];
                        }
                        else if (i > vertex)
                        {
                            newMatrix[i + 1, j] = matrix[i, j];
                        }
                        else if (j > vertex)
                        {
                            newMatrix[i, j + 1] = matrix[i, j];
                        }
                        else
                        {
                            newMatrix[i, j] = matrix[i, j];
                        }
                    }
                }
            }

          // Следующий цикл for используется для копирования ребер из разделенной вершины в новые вершины:
            for (int i = 0; i < size; i++)
            {
                if (i != vertex)
```

```
                {
                    newMatrix[vertex, i] = matrix[vertex, i];
                    newMatrix[i, vertex] = matrix[i, vertex];
                }
            }
         // обновление исходной матрицы matrix новой матрицей newMatrix,
которая содержит добавленную вершину и соответствующие ребра.
            matrix = newMatrix;
        }

    }
}
```

Описание кода программы по заданию 3:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace _3
{
    internal class Program
    {
        //Метод GenerateAdjacencyMatrix генерирует случайную матрицу
смежности для графа заданного размера.
        private static int[,] GenerateAdjacencyMatrix(int size, Random
rand)
        {

            int[,] matrix = new int[size, size];

            for (int i = 0; i < size; i++)
            {
                for (int j = 0; j < size; j++)
                {
                    if (i != j)
                    {
                        matrix[i, j] = rand.Next(2);
                        matrix[j, i] = matrix[i, j];
                    }
                }
            }
            return matrix;
        }
        // Метод для вывода матрицы на экран
        static void PrintMatrix(int[,] matrix)
        {
            int size = matrix.GetLength(0);

            for (int i = 0; i < size; i++)
            {
```

```
                for (int j = 0; j < size; j++)
                {
                    Console.Write(matrix[i, j] + " ");
                }
                Console.WriteLine();
            }
            Console.WriteLine();
        }
        //Метод Union выполняет операцию объединения
        private static int[,] Union(int[,] matrix1, int[,] matrix2)
        {
            //Метод Union создает новую матрицу размером, равным максимальному размеру из matrix1 и matrix2
            int size = Math.Max(matrix1.GetLength(0), matrix2.GetLength(0));
            int[,] result = new int[size, size];

            //Затем он перебирает все элементы новой матрицы и устанавливает значения, используя значения из matrix1 и matrix2
            for (int i = 0; i < size; i++)
            {
                for (int j = 0; j < size; j++)
                {
               //Если элемент существует в обоих матрицах, то значение в новой матрице будет равно 1, иначе 0.
               if (i < matrix1.GetLength(0) && j < matrix1.GetLength(1))
                    {
                        result[i, j] = matrix1[i, j];
                    }
               if (i < matrix2.GetLength(0) && j < matrix2.GetLength(1))
                    {
                        result[i, j] |= matrix2[i, j];
                    }
                }
            }

            return result;
        }

        //Метод Intersection выполняет операцию пересечения
      private static int[,] Intersection(int[,] matrix1, int[,] matrix2)
        {
            //Метод Intersection также создает новую матрицу размером, равным максимальному размеру из matrix1 и matrix2
            int size = Math.Max(matrix1.GetLength(0), matrix2.GetLength(0));
            int[,] result = new int[size, size];

            //Затем он перебирает все элементы новой матрицы и устанавливает значения, используя значения из matrix1 и matrix2
            for (int i = 0; i < size; i++)
```

```
            {
                for (int j = 0; j < size; j++)
                {
               //Если элемент существует в обоих матрицах, то значение в новой матрице будет равно 1, иначе 0.
                if (i < matrix1.GetLength(0) && j < matrix1.GetLength(1) && i < matrix2.GetLength(0) && j < matrix2.GetLength(1))
                    {
                        result[i, j] = matrix1[i, j] & matrix2[i, j];
                    }
                }
            }

            return result;
        }

        //Метод RingSum выполняет операцию кольцевой суммы
        private static int[,] RingSum(int[,] matrix1, int[,] matrix2)
        {
            //Объявление переменной size и присваивание ей значения максимального измерения(строк или столбцов) между matrix1 и matrix2, используя метод Math.Max.
            int size = Math.Max(matrix1.GetLength(0), matrix2.GetLength(0));
             //создает новую матрицу размером, равным максимальному размеру из matrix1 и matrix2
            int[,] result = new int[size, size];

            //Затем он перебирает все элементы новой матрицы и устанавливает значения, используя значения из matrix1 и matrix2.
            for (int i = 0; i < size; i++)
            {
                for (int j = 0; j < size; j++)
                {
               //Если элемент существует только в одной из матриц, то значение в новой матрице будет равно 1, иначе 0.
                if (i < matrix1.GetLength(0) && j < matrix1.GetLength(1))
                    {
                        result[i, j] = matrix1[i, j];
                    }
                if (i < matrix2.GetLength(0) && j < matrix2.GetLength(1))
                    {
                        //Применение операции побитового исключающего ИЛИ (^=) к элементу массива result[i, j] и элементу массива matrix2[i, j].
                   //Результат присваивается обратно элементу массива result[i, j].
                        result[i, j] ^= matrix2[i, j];
                    }
                }
            }
```

```
            return result;
        }

        // Пример использования
        static void Main(string[] args)
        {
            Console.Write("Введите размер 1 матрицы: ");
            int sizeOne = Convert.ToInt32(Console.ReadLine());
            Console.Write("Введите размер 2 матрицы: ");
            int sizeTwo = Convert.ToInt32(Console.ReadLine());

            Random rand1 = new Random();
            Random rand2 = new Random();

            int[,] matrix1 = GenerateAdjacencyMatrix(sizeOne, rand1);
            int[,] matrix2 = GenerateAdjacencyMatrix(sizeTwo, rand2);

            Console.WriteLine("Матрица G1:");
            PrintMatrix(matrix1);

            Console.WriteLine("Матрица G2:");
            PrintMatrix(matrix2);

            Console.WriteLine("Объединение G1 и G2:");
            int[,] unionMatrix = Union(matrix1, matrix2);
            PrintMatrix(unionMatrix);

            Console.WriteLine("Пересечение G1 и G2:");
         int[,] intersectionMatrix = Intersection(matrix1, matrix2);
            PrintMatrix(intersectionMatrix);

            Console.WriteLine("Кольцевая сумма G1 и G2:");
            int[,] ringSumMatrix = RingSum(matrix1, matrix2);
            PrintMatrix(ringSumMatrix);
        }

    }
}
```

Описание кода программы по заданию 4:

```
using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;


namespace _4
```

```
{
    internal class Program
    {
        static void Main(string[] args)
        {
            Console.Write("Введите размер 1 матрицы: ");
            int sizeOne = Convert.ToInt32(Console.ReadLine());
            Console.Write("Введите размер 2 матрицы: ");
            int sizeTwo = Convert.ToInt32(Console.ReadLine());

            Random rand1 = new Random();
            Random rand2 = new Random();

            int[,] matrix1 = GenerateAdjacencyMatrix(sizeOne, rand1);
            int[,] matrix2 = GenerateAdjacencyMatrix(sizeTwo, rand2);

            Console.WriteLine("Матрица G1:");
            PrintMatrix(matrix1);

            Console.WriteLine("Матрица G2:");
            PrintMatrix(matrix2);

            int[,] resultMatrix = MultiplyMatrices(matrix1, matrix2);

            Console.WriteLine("Матрица G = G1 X G2:");
            PrintMatrix(resultMatrix);

        }
        //Метод MultiplyMatrices принимает две матрицы смежности
matrix1 и matrix2 и возвращает новую матрицу, которая является
декартовым произведением этих двух матриц.
```

```
        static int[,] MultiplyMatrices(int[,] matrix1, int[,] matrix2)
        {
            int sizeOne = matrix1.GetLength(0);
            int sizeTwo = matrix2.GetLength(1);

            //Размер новой матрицы определяется как произведение размеров matrix1 и matrix2.
            int[,] matrix = new int[sizeOne* sizeTwo, sizeOne* sizeTwo];

            //Внутри метода используются четыре вложенных цикла for для перебора всех возможных комбинаций вершин из matrix1 и matrix2
            for (int i = 0; i < sizeOne; i++)
            {
                for (int j = 0; j < sizeOne; j++)
                {
                    for (int k = 0; k < sizeTwo; k++)

                    {
                        for (int l = 0; l < sizeTwo; l++)
                        {
                            //Если в обоих матрицах соответствующие вершины имеют значение 1, то в новой матрице соответствующее ребро устанавливается в значение 1.
                            if (matrix1[i, j] == 1 && matrix2[k, l] == 1)
                    {
                matrix[i* sizeTwo +k, j* sizeTwo +l] = 1;
            }
        }
    }
}
```

```
}


    return matrix;
}
        //Метод GenerateAdjacencyMatrix генерирует случайную матрицу смежности для графа заданного размера
        private static int[,] GenerateAdjacencyMatrix(int size, Random rand)
        {

            int[,] matrix = new int[size, size];

            for (int i = 0; i < size; i++)
            {
                for (int j = 0; j < size; j++)
                {
                    if (i != j)
                    {
                        matrix[i, j] = rand.Next(2);
                        matrix[j, i] = matrix[i, j];
                    }
                }
            }
            return matrix;
        }
        // Метод для вывода матрицы на экран
        static void PrintMatrix(int[,] matrix)
        {
            int size = matrix.GetLength(0);

            for (int i = 0; i < size; i++)
```

```
            {
                for (int j = 0; j < size; j++)
                {
                    Console.Write(matrix[i, j] + " ");
                }
                Console.WriteLine();
            }
            Console.WriteLine();
        }
    }
}
```

Результат работы программы 1:

Результат работы программы 2(1часть):

Результат работы программы 3:

Результат работы программы 4:

Вывод: в ходе лабораторной работы были получены навыки работы с матричной формой представления графа. Были реализованы такие операции как: отождествление вершин, стягивание ребра, расщепление вершины. А также: объединения графов, пересечения графов, кольцевой суммы графов и ещё операция декартова произведения графов.