МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ
ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Кафедра «Вычислительная техника»

ОТЧЕТ

По лабораторной работе №10

«Поиск расстояний в графе»

По дисциплине «Л и ОА в ИЗ»

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

Приняли: Юрова О.В.
Акифьев И.В.

2023

Цель: найти центр тяжести графа.

Объяснение кода:

```
using System;

using System.Collections.Generic;

using System.Linq;


class Program

{

    static void Main(string[] args)

    {

        Console.Write("Введите размер матрицы: ");

        int size;

        while (!int.TryParse(Console.ReadLine(), out size))

        {

            Console.WriteLine("Введите целое число");

        }

        int[,] adjacencyMatrix = GenerateAdjacencyMatrix(size);

        Console.WriteLine("Матрица смежности для неориентированного
графа:");

        PrintMatrix(adjacencyMatrix);

        int[] sums = FindSumDistances(adjacencyMatrix, size);

        Console.WriteLine("Расстояния от каждой вершины:");

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

        {

            Console.WriteLine("Вершина " + (i + 1) + ": " + sums[i]);

        }


        // Находим минимальное значение суммы расстояний

        int minSum = sums.Min();
```

```
        Console.Write("\n" + "Минимальное расстояние: " + minSum +
"\n");


        // Находим все вершины с минимальным расстоянием


        //Создается пустой список centroidVertices для хранения
вершин, которые являются центром тяжести.
        List<int> centroidVertices = new List<int>();


        //Затем, проходим циклом по всем вершинам графа
        for (int i = 0; i < size; i++)
        {
            //проверяем если сумма расстояний от текущей вершины до
остальных равна минимальному расстоянию (minSum).
            if (sums[i] == minSum)
            {
                //то добавляем эту вершину в список centroidVertices.
                centroidVertices.Add(i + 1);
            }
        }
        Console.WriteLine("Центр тяжести графа: ");


        //выводим каждую вершину из списка centroidVertices на
отдельной строке.
        foreach (int vertex in centroidVertices)
        {
            Console.WriteLine(vertex + " ");
        }


    }
    //Генерация матрицы смежности неориентированного графа
    private static int[,] GenerateAdjacencyMatrix(int size)
```

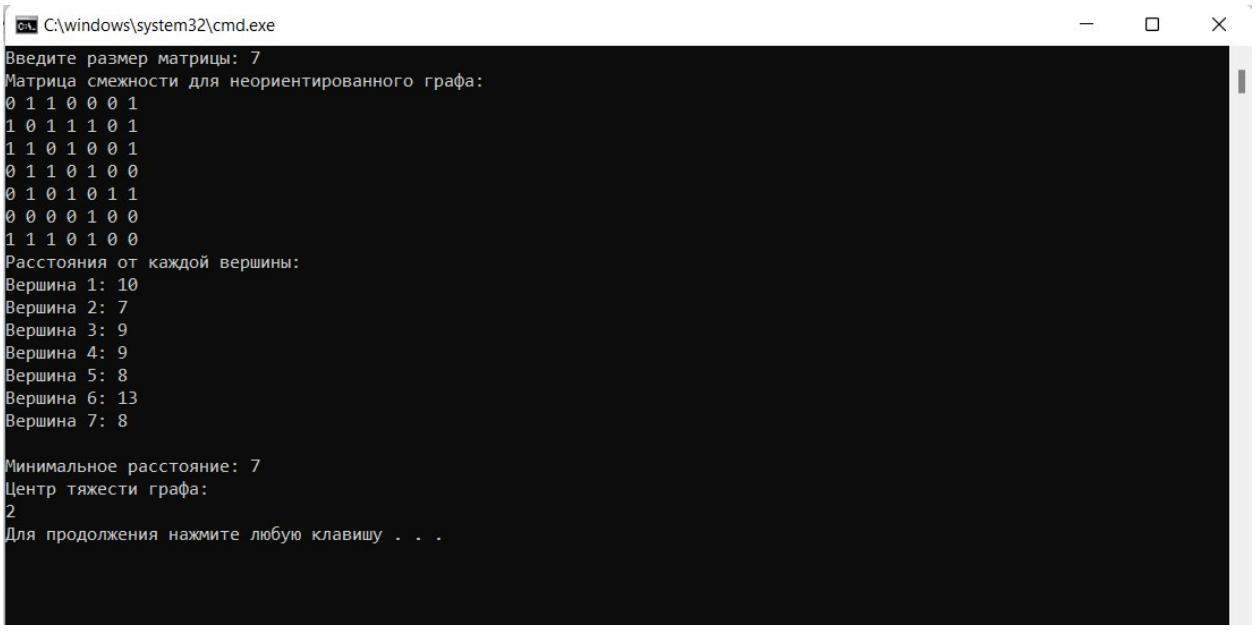
```
{
    Random r = new Random();
    int[,] matrix = new int[size, size];
    for (int i = 0; i < size; i++)
    {
        for (int j = 0; j < size; j++)
        {
            if (i != j)
            {
                matrix[i, j] = r.Next(2);
                matrix[j, i] = matrix[i, j];
         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

```
    private static int[] FindSumDistances(int[,] adjacencyMatrix, int
size)

    {

        //Создаем массив sums размером size для хранения сумм
расстояний от каждой вершины.

        int[] sums = new int[size];

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

        {

            int[] distances = FindDistancesFromVertex(adjacencyMatrix,
i, size);



            //Суммируем все значения в полученном массиве расстояний с
помощью метода Sum() и сохраняем результат в sums[i]

            sums[i] = distances.Sum();

        }

        //После завершения цикла возвращаем массив sums, содержащий
суммы расстояний от каждой вершины.

        return sums;

    }



    private static int[] FindDistancesFromVertex(int[,]
adjacencyMatrix, int vertex, int size)

    {

        //Создаем массив distances размером size и инициализируем его
значениями int.MaxValue.

        //Это будет массив, в котором каждый элемент будет
представлять расстояние от текущей

        //вершины до соответствующей вершины с индексом i.

        int[] distances = Enumerable.Repeat(int.MaxValue,
size).ToArray();



        //Устанавливаем distances[vertex] в 0, так как расстояние от
вершины до самой себя равно 0.

        distances[vertex] = 0;
```

```
        //Создаем очередь queue для хранения вершин, которые будут обрабатываться.
        Queue<int> queue = new Queue<int>();


        //Добавляем текущую вершину vertex в очередь queue с помощью метода Enqueue().
        queue.Enqueue(vertex);


        // Запускаем цикл while, который будет выполняться, пока очередь queue не станет пустой.
        while (queue.Count > 0)
        {
            //Внутри цикла извлекаем текущую вершину currentVertex из очереди queue с помощью метода Dequeue().
            int currentVertex = queue.Dequeue();
            for (int i = 0; i < size; i++)
            {
                //Проверяем, есть ли ребро между текущей вершиной currentVertex
                //и вершиной с индексом i путем проверки значения adjacencyMatrix[currentVertex, i].
                //Если значение равно 1 и расстояние до вершины i
                //равно int.MaxValue (что означает, что эта вершина еще не была посещена),
                //то выполняем следующий блок кода.
                if (adjacencyMatrix[currentVertex, i] == 1 && distances[i] == int.MaxValue)
                {
                    //Устанавливаем расстояние до вершины i равным расстоянию до
                    //текущей вершины currentVertex плюс 1 (distances[currentVertex] + 1),
```

```
                    //так как мы переходим к следующей вершине через
текущую.
                    distances[i] = distances[currentVertex] + 1;


                    //Добавляем вершину i в очередь queue с помощью
метода Enqueue(), чтобы она была обработана на следующей итерации
цикла while.
                    queue.Enqueue(i);
                }
            }
        }
        //возвращаем массив distances, содержащий расстояния от
текущей вершины до каждой вершины.
        return distances;
    }
}
```

Результат работы программы:



Вывод: научились находить центр тяжести графа в неориентированном, невзвешенном графе.