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 № 3

по дисциплине «Теория множеств и алгоритмы на графах»

по теме: «Матричные способы представления графов»

Выполнил: Ольховский Н.С., ИТА-123

Проверила: Кузьмина Т.М.

Москва, 2025

# Вариант 13

# Задание

На форму нужно поместить 2 таблицы: одну для матрицы смежности, другую для матрицы инцидентности. Одна из таблиц предназначена для ввода значений пользователем, другая для вывода расчетных данных. Кроме таблиц на форме должны быть компонент для вывода рисунка графа и кнопки управления.

Поскольку количество вершин фиксировано и определяется вариантом задания, то вершины графа можно нарисовать при запуске программы. Пользователь задает только ребра графа. У студентов с чётными номерами варианта задания в качестве входной матрицы должна быть матрица смежности графа. У студентов с **нечётными номерами - матрица инцидентности**. Каждый студент работает с графом, у которого количество вершин вычисляется по формуле (n/2)+4, где n-вариант задания.

Пользователь задаёт матрицу смежности (инцидентности) графа. Программа должна проверить корректность введённой матрицы. Если пользователь ошибётся, то программа должна сообщить об этом. Построить матрицу инцидентности (смежности) графа и вывести рисунок графа на экран.

# Работа программы

Скриншот работы представлен на рисунке 1 и 2.

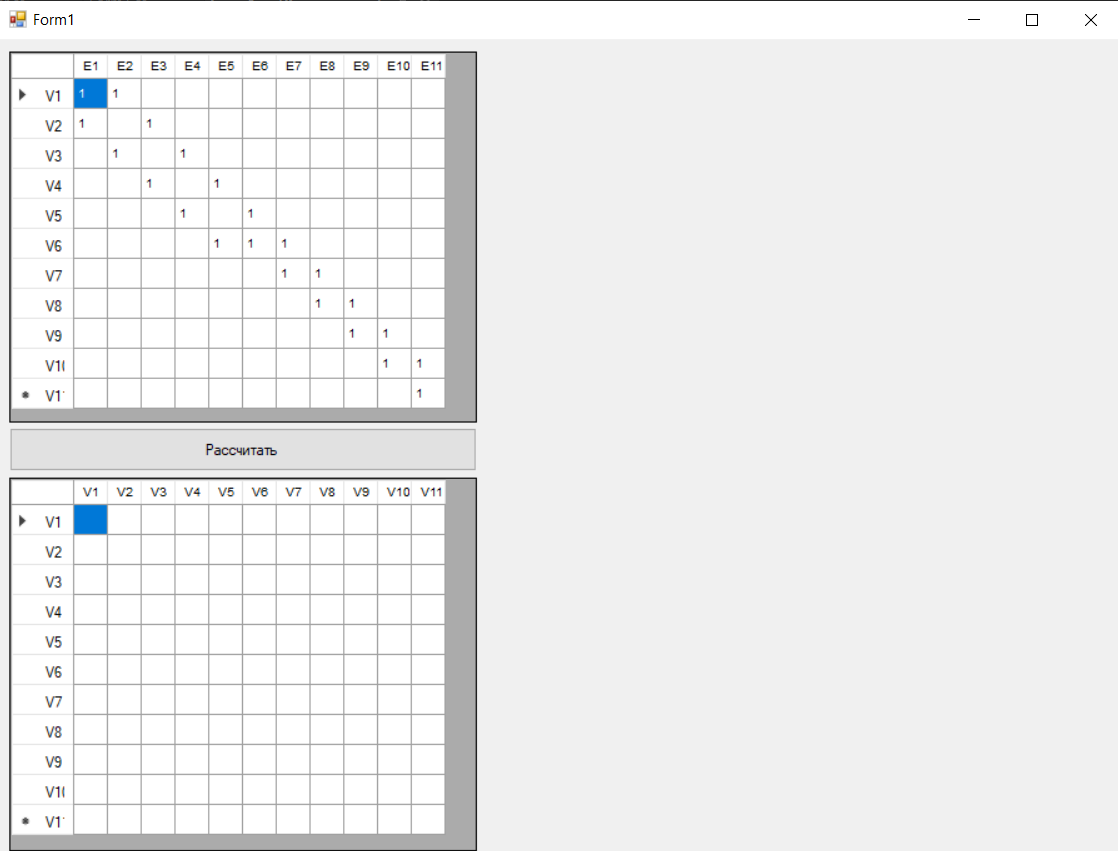


Рис. 1 – Начальное состояние программы

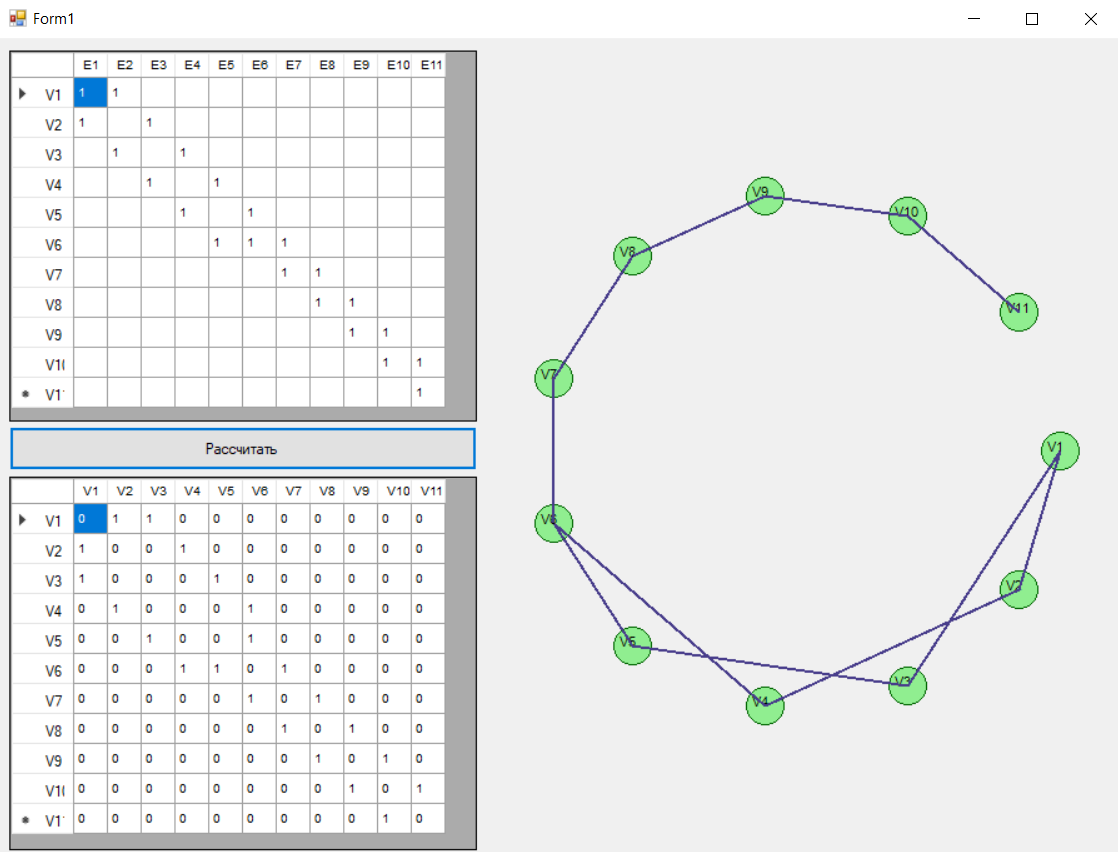


Рис. 2 – Результат расчёта

# Текст программы

using System;

using System.Drawing;

using System.Windows.Forms;

namespace Lab3Form

{

public partial class Form1 : Form

{

const int VertexCount = 11;

const int MaxEdges = 11;

private void Form1\_Load(object sender, EventArgs e)

{

}

public Form1()

{

InitializeComponent();

InitGrids();

}

private void InitGrids()

{

// Таблица инцидентности

dataGridViewIncidence.ColumnCount = MaxEdges;

dataGridViewIncidence.RowCount = VertexCount;

dataGridViewIncidence.RowTemplate.Height = 12;

dataGridViewIncidence.ColumnHeadersDefaultCellStyle.Font = new Font("Arial", 7);

dataGridViewIncidence.DefaultCellStyle.Font = new Font("Arial", 7);

foreach (DataGridViewColumn col in dataGridViewIncidence.Columns)

{

col.Width = 27;

}

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

dataGridViewIncidence.Columns[i].HeaderText = $"E{i + 1}";

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

dataGridViewIncidence.Rows[i].HeaderCell.Value = $"V{i + 1}";

// Данные по умолчанию

var defaultEdges = new (int, int)[]

{

(0, 1), (0, 2), (1, 3), (2, 4), (3, 5),

(4, 5), (5, 6), (6, 7), (7, 8), (8, 9), (9, 10)

};

for (int e = 0; e < defaultEdges.Length; e++)

{

dataGridViewIncidence.Rows[defaultEdges[e].Item1].Cells[e].Value = 1;

dataGridViewIncidence.Rows[defaultEdges[e].Item2].Cells[e].Value = 1;

}

// Таблица смежности

dataGridViewAdjacency.ColumnCount = VertexCount;

dataGridViewAdjacency.RowCount = VertexCount;

dataGridViewAdjacency.RowTemplate.Height = 12;

dataGridViewAdjacency.ColumnHeadersDefaultCellStyle.Font = new Font("Arial", 7);

dataGridViewAdjacency.DefaultCellStyle.Font = new Font("Arial", 7);

foreach (DataGridViewColumn col in dataGridViewAdjacency.Columns)

{

col.Width = 27;

}

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

{

dataGridViewAdjacency.Columns[i].HeaderText = $"V{i + 1}";

dataGridViewAdjacency.Rows[i].HeaderCell.Value = $"V{i + 1}";

}

DrawGraph();

}

private void DrawGraph(int[,] adjacency = null)

{

Graphics g = panelGraph.CreateGraphics();

g.Clear(panelGraph.BackColor);

Point[] positions = new Point[VertexCount];

int centerX = panelGraph.Width / 2;

int centerY = panelGraph.Height / 2;

int radius = Math.Min(centerX, centerY) - 40;

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

{

double angle = 2 \* Math.PI \* i / VertexCount;

int x = centerX + (int)(radius \* Math.Cos(angle));

int y = centerY + (int)(radius \* Math.Sin(angle));

positions[i] = new Point(x, y);

g.FillEllipse(Brushes.LightGreen, x - 15, y - 15, 30, 30);

g.DrawEllipse(Pens.DarkGreen, x - 15, y - 15, 30, 30);

g.DrawString($"V{i + 1}", new Font("Arial", 8), Brushes.Black, x - 12, y - 10);

}

if (adjacency != null)

{

Pen edgePen = new Pen(Color.DarkSlateBlue, 2);

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

for (int j = i + 1; j < VertexCount; j++)

if (adjacency[i, j] == 1)

g.DrawLine(edgePen, positions[i], positions[j]);

}

}

private void buttonCalculate\_Click\_1(object sender, EventArgs e)

{

try

{

int[,] incidence = new int[VertexCount, MaxEdges];

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

{

for (int j = 0; j < MaxEdges; j++)

{

var value = dataGridViewIncidence.Rows[i].Cells[j].Value;

incidence[i, j] = value == null || value.ToString() == "" ? 0 : int.Parse(value.ToString());

if (incidence[i, j] != 0 && incidence[i, j] != 1)

throw new Exception("Допустимы только значения 0 или 1.");

}

}

// Проверка: в каждом столбце должно быть ровно две 1

for (int j = 0; j < MaxEdges; j++)

{

int countOnes = 0;

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

if (incidence[i, j] == 1)

countOnes++;

if (countOnes != 2)

{

MessageBox.Show($"Ошибка в ребре E{j + 1}: должно быть ровно две 1.");

return;

}

}

// Строим матрицу смежности

int[,] adjacency = new int[VertexCount, VertexCount];

for (int j = 0; j < MaxEdges; j++)

{

int a = -1, b = -1;

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

{

if (incidence[i, j] == 1)

{

if (a == -1) a = i;

else b = i;

}

}

if (a != -1 && b != -1)

{

adjacency[a, b] = 1;

adjacency[b, a] = 1;

}

}

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

for (int j = 0; j < VertexCount; j++)

dataGridViewAdjacency.Rows[i].Cells[j].Value = adjacency[i, j];

DrawGraph(adjacency);

}

catch (Exception ex)

{

MessageBox.Show("Ошибка: " + ex.Message);

}

}

}

}