МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение высшего профессионального образования

«Тихоокеанский государственный университет»

Кафедра «Программное обеспечение вычислительной техники и автоматизированных систем»

«Изучение и реализация алгоритмов дискретной математике.»

«Поиск кратчайшего пути с помощью алгоритма Беллмана–Форда.»

Текстовый документ курсовой работы

по дисциплине «Алгоритмы дискретной математики»

КР. 180008865.ТД

Выполнил студент Пшеничный Д. О.

Факультет, группа ФКФН, ПО(аб) – 81

Руководитель работы Бахрушина Г.И.

Виза: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(доработать, к защите и т.д.)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Хабаровск – 2020г.

СОДЕРЖАНИЕ

[ВВЕДЕНИЕ 3](#_Toc40689235)

[1 Постановка задачи (в соответствии с вариантом) 4](#_Toc40689236)

[2 Описание алгоритма. 5](#_Toc40689237)

[3 Описание программы 6](#_Toc40689238)

[3.1 Назначение программы 6](#_Toc40689239)

[3.2 Язык и среда программирования 6](#_Toc40689240)

[3.3 Описание структур данных и функций 6](#_Toc40689241)

[ЗАКЛЮЧЕНИЕ 8](#_Toc40689242)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 9](#_Toc40689243)

[ПРИЛОЖЕНИЕ А 10](#_Toc40689244)

[ПРИЛОЖЕНИЕ Б 22](#_Toc40689245)

## ВВЕДЕНИЕ

В ходе курсовой работы необходимо реализовать поиск матрицы достижимости ориентированного графа.

Матрица достижимости простого ориентированного графа G = (V, A) —  бинарная матрица замыкания по транзитивности отношения A (оно задаётся матрицей смежности графа). Таким образом, в матрице достижимости хранится информация о существовании путей между вершинами орграфа. В работе также используются матрица смежности и матрица инцидентности.

Матрица смежности графа G с конечным числом вершин n (пронумерованных числами от 1 до n) — это квадратная матрица A размера n, в которой значение элемента aij равно числу рёбер из i-й вершины графа в j-ю вершину.

Иногда, особенно в случае неориентированного графа, петля (ребро из i-й вершины в саму себя) считается за два ребра, то есть значение диагонального элемента aii в этом случае равно удвоенному числу петель вокруг i-й вершины.

Матрица смежности простого графа (не содержащего петель и кратных рёбер) является бинарной матрицей и содержит нули на главной диагонали.

Матрица инцидентности — одна из форм представления графа, в которой указываются связи между инцидентными элементами графа (ребро(дуга) и вершина). Столбцы матрицы соответствуют ребрам, строки — вершинам. Ненулевое значение в ячейке матрицы указывает связь между вершиной и ребром (их инцидентность).

В случае ориентированного графа каждой дуге <x, y> ставится в соответствующем столбце: «1» в строке вершины x и «-1» в строке вершины y; если связи между вершиной и ребром нет, то в соответствующую ячейку ставится «0».

# Постановка задачи (в соответствии с вариантом)

Осуществить поиск матрицы достижимости для ориентированного графа. Пользователь строит граф, используя предложенные инструменты. Также реализовать возможность импорта и экспорта графа в текстовый файл. Программа должна выводить матрицы достижимости, смежности и инцидентности.

# Описание алгоритма.

Для нахождения матрицы достижимости воспользуемся способом перемножения матриц. Пусть дан граф G = (V, A), матрица смежности которого есть A. Матрица смежности даёт информацию о всех путях длины 1, то есть рёбрах, в орграфе. Для поиска путей длины 2 можно найти композицию отношения A с самим собой. По определению, данная матрица композиции отношений есть A2. Для путей длины k справедливо Ak. /3-5/

Для получения матрицы достижимости с количеством путей, нужно сложить полученные композиции. Матрица достижимости R(G) = E + A + A2 + A3 + … + Ak

Воспользуемся функцией Main.CreateAviMatrix(). В неё создаётся двумерный массив res[,], в который будут записываться результаты возведения в степень матрицы смежности, полученной с помощью функции Main.CreateAdjMatrix(). В результате получим матрицу, отображающую количество возможных путей из одной любой точки в другую любую точку.

# Описание программы.

## 3.1 Назначение программы

Разработанная программа представляет собой визуальное приложение по созданию графа и нахождению матрицы достижимости. Программа также имеет функции импорта и экспорта графа в текстовый файл, вывода матриц смежности и инцидентности. Функционирование программы и код функций представлены в приложении А и Б соответственно.

## 3.2 Язык и среда программирования

Программа была создана на языке C# с использованием среды программирования (IDE) Visual Studio 2019 Community. /1-2/

## 3.3 Описание структур данных и функций

**Class Form1** – класс главной формы

**class Dot –** класс точки

**class Line** – класс линии

**class Main** – класс для хранения некоторых данных и функций

**void Main.CreateDot(int n)** – генерирует точки для импортированного графа

**bool Main.CheckLineAviability(Dot firstDot, Dot secondDon)** – проверяет, нет ли уже такое ребро

**Dot Main.FindByInd(int ind)** – возвращает экземпляр точки по индексу

**int[,] Main.CreateIncMatrix()** – создаёт и возвращает матрицу инцидентности для заданного графа

**int[,] Main.CreateAdjMatrix()** – создаёт и возвращает матрицу смежности для заданного графа

**int[,] Main.CreareAviMatrix()** – создаёт и возвращает матрицу достижимости для заданного графа

**int[,] Main.Multiplication(int[,] a, int[,] b)** – умножает матрицу a на матрицу b

**void Form1.RemoveDot(int ind)** – удаляет точку по индексу

# ЗАКЛЮЧЕНИЕ

Выполнение курсовой работы завершилось созданием программного продукта, который обладает функциональными возможностями, определенными заданием. Для разработки программного продукта был применен визуальный способ создания ориентированного графа.

В приложении А представлен код программы. В приложении Б приведены скриншоты, полученные в результате тестирования программы.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. <https://metanit.com> – ресурс посвященный различным языкам и технологиям программирования, в частности C#
2. Бен Албахари и Джозеф Албахари. C# 7.0 in a Nutshell: The Definitive Reference, 28 октября 2017 г.
3. <http://poivs.tsput.ru/ru/Math/DiscreteMath/GraphTheory/AccessibilityMatrix> - Матрица достижимости, перемножение матриц
4. Джеймс Андерсон. Дискретная математика и комбинаторика, 2004
5. Акимов О.Е. Дискретная математика. Логика. Группы. Графы. 2001.

# ПРИЛОЖЕНИЕ А

Код программы

Содержимое файла Form1.cs

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using System.IO;

namespace GrafBuilder

{

public partial class Form1 : Form

{

Pen[] pen = new Pen[] {

new Pen(Color.DarkGoldenrod, 2),

new Pen(Color.Black, 2),

new Pen(Color.Red, 2),

new Pen(Color.Blue, 2),

new Pen(Color.Green, 2),

new Pen(Color.Purple, 2),

new Pen(Color.Yellow, 2),

new Pen(Color.Pink, 2),

new Pen(Color.Violet, 2),

new Pen(Color.Brown, 2),

new Pen(Color.Magenta, 2) };

int[,] matrix;

public Form1()

{

InitializeComponent();

}

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

{

}

private void Add\_dot\_Click(object sender, EventArgs e)

{

try

{

int x = int.Parse(textBox1.Text);

int y = int.Parse(textBox2.Text);

Main.dotCount++;

Main.dots.Add(new Dot(x, y, Main.dotCount));

}

catch

{

MessageBox.Show("Введены неверные значения!!!");

}

RefreshForm();

}

void RefreshForm()

{

panel1.Invalidate();

if (Main.dotCount == Main.dotMax)

add\_dot.Enabled = !(Main.dotCount == Main.dotMax);

string dotList = "";

foreach (Dot x in Main.dots)

{

dotList += x.DotInd.ToString() + ". x:" + x.DotX.ToString() + " y:" + x.DotY.ToString() + "\n";

}

string lineList = "";

foreach(Line x in Main.lines)

{

lineList += (x.line\_ind + 1).ToString() + ". " + x.dot1.DotInd.ToString() + " -> " + x.dot2.DotInd.ToString() + "\n";

}

label4.Text = lineList;

label1.Text = dotList;

}

private void Panel1\_Paint(object sender, PaintEventArgs e)

{

Graphics graphics = panel1.CreateGraphics();

foreach(Line x in Main.lines) // Отрисовка линий

{

if (x.dot1 != x.dot2)

graphics.DrawLine(pen[x.dot2.DotInd], x.dot1.DotX, x.dot1.DotY, x.dot2.DotX, x.dot2.DotY);

else

graphics.DrawEllipse(pen[x.dot2.DotInd], x.dot1.DotX, x.dot1.DotY, 30f, 30f);

}

foreach(Dot x in Main.dots) // Отрисовка точек

{

graphics.DrawRectangle(pen[x.DotInd], x.DotX - 2, x.DotY - 2, 5, 5);

}

}

private void Add\_line\_Click(object sender, EventArgs e)

{

try

{

int first = int.Parse(textBox3.Text);

int second = int.Parse(textBox4.Text);

if (Main.CheckLineAviability(Main.FindByInd(first), Main.FindByInd(second)))

{

if (first > 0 && first <= Main.dotCount && second > 0 && second <= Main.dotCount )// && first != second)

{

Main.lineCount++;

Main.lines.Add(new Line(Main.FindByInd(first), Main.FindByInd(second), Main.lineCount));

}

else

{

MessageBox.Show("Введены неверные значения!!!");

}

}

else

{

MessageBox.Show("Такое ребро уже существует!");

}

}

catch

{

MessageBox.Show("Введены неверные значения!!!");

}

RefreshForm();

}

private void Panel1\_MouseDown(object sender, MouseEventArgs e)

{

foreach(Dot x in Main.dots)

{

if (e.X <= x.DotX + 4 && e.X >= x.DotX - 4 && e.Y <= x.DotY + 4 && e.Y >= x.DotY - 4)

{

Main.activeDotInd = x.DotInd;

RefreshForm();

break;

}

}

}

private void Panel1\_MouseMove(object sender, MouseEventArgs e)

{

Main.mouseX = e.X;

Main.mouseY = e.Y;

if (Main.activeDotInd != 0)

{

Dot temp = Main.FindByInd(Main.activeDotInd);

temp.DotX = e.X;

temp.DotY = e.Y;

RefreshForm();

}

}

private void Panel1\_MouseUp(object sender, MouseEventArgs e)

{

if (Main.activeDotInd != 0)

{

Main.activeDotInd = 0;

RefreshForm();

}

}

private void Panel1\_MouseDoubleClick(object sender, MouseEventArgs e)

{

if (add\_dot.Enabled)

{

bool b = true;

for (int i = 0; i < Main.dots.Count(); i++)

{

Dot temp = Main.dots[i];

if (e.X <= temp.DotX + 4 && e.X >= temp.DotX - 4 && e.Y <= temp.DotY + 4 && e.Y >= temp.DotY - 4)

{

b = false;

break;

}

}

if (b)

{

Main.dotCount++;

Main.dots.Add(new Dot(e.X, e.Y, Main.dotCount));

RefreshForm();

}

}

}

private void Clear\_button\_Click(object sender, EventArgs e)

{

Clear();

}

private void Remove\_button\_Click(object sender, EventArgs e)

{

try

{

int ind = int.Parse(textBox5.Text);

RemoveDot(ind);

add\_dot.Enabled = true;

}

catch

{

MessageBox.Show("Введено неверное значение!!!");

}

}

private void Clear()

{

Main.dots.Clear();

Main.lines.Clear();

Main.dotCount = 0;

Main.lineCount = 0;

RefreshForm();

}

private void RemoveDot(int ind)

{

for (int i = 0; i < Main.lines.Count(); i++)

{

Line temp = Main.lines[i];

if (temp.dot1.DotInd == ind || temp.dot2.DotInd == ind)

{

Main.lines.Remove(Main.lines[i]);

Main.lineCount--;

i--;

}

}

for (int i = 0; i < Main.lines.Count(); i++)

{

Main.lines[i].line\_ind = i + 1;

}

for (int i = 0; i < Main.dots.Count(); i++)

{

Dot temp = Main.dots[i];

if (temp.DotInd == ind)

{

Main.dotCount--;

Main.dots.Remove(Main.dots[i]);

break;

}

}

for (int i = 0; i < Main.dots.Count(); i++)

{

Main.dots[i].DotInd = i + 1;

}

RefreshForm();

}

private void exportButton\_Click(object sender, EventArgs e)

{

var newM = Main.CreateIncMatrix();

string res = "";

res += Main.dots.Count.ToString() + " " + Main.lines.Count.ToString() + "\n";

for (int i = 0; i < Main.dots.Count; i++)

{

for (int j = 0; j < Main.lines.Count; j++)

{

if (j != Main.lines.Count - 1)

res += newM[i, j] + " ";

else

res += newM[i, j];

}

res += '\n';

}

Form3 exportForm = new Form3();

exportForm.Owner = this;

exportForm.ShowDialog();

if(exportForm.DialogResult == DialogResult.OK)

{

var exportFile = new StreamWriter(Main.destPath, false);

exportFile.Write(res);

exportFile.Close();

}

}

private void importButton\_Click(object sender, EventArgs e)

{

string fileText = "";

Form2 importForm = new Form2();

importForm.Owner = this;

importForm.ShowDialog();

if (importForm.DialogResult == DialogResult.OK)

{

Clear();

StreamReader file = new StreamReader(Main.sourcePath);

try

{

fileText = file.ReadToEnd();

}

catch

{

MessageBox.Show("Введён неверный путь к файлу!");

return;

}

file.Close();

Main.sourcePath = "";

int n = 0;

int m = 0;

try

{

string[] splittedText = fileText.Split(new char[] {'\n', ' '});

string text = "";

n = int.Parse(splittedText[0]);

m = int.Parse(splittedText[1]);

matrix = new int[n, m];

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

{

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

{

matrix[i, j] = int.Parse(splittedText[2 + i \* m + j]);

text += matrix[i, j].ToString() + " ";

}

text += '\n';

}

MessageBox.Show(text);

Main.CreateDots(n);

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

{

int ind1 = -1;

int ind2 = -1;

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

{

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

{

ind1 = i + 1;

}

if (matrix[i, j] == -1)

{

ind2 = i + 1;

}

}

if(ind1 != -1 && ind2 != -1)

Main.lines.Add(new Line(Main.FindByInd(ind1), Main.FindByInd(ind2), Main.lineCount));

else

Main.lines.Add(new Line(Main.FindByInd(ind2), Main.FindByInd(ind2), Main.lineCount));

Main.lineCount++;

}

RefreshForm();

}

catch

{

MessageBox.Show("Файл не прочитан");

}

}

}

private void label2\_Click(object sender, EventArgs e)

{

}

private void incMatrixShow\_Click(object sender, EventArgs e)

{

var newM = Main.CreateIncMatrix();

string res = "";

for (int i = 0; i < Main.dots.Count; i++)

{

for (int j = 0; j < Main.lines.Count; j++)

{

if(newM[i, j] == -1)

{

if (j != Main.lines.Count - 1)

{

res += newM[i, j] + " ";

}

else

{

res += newM[i, j];

}

}

else

{

if (j != Main.lines.Count - 1)

{

res += " " + newM[i, j] + " ";

}

else

{

res += " " + newM[i, j];

}

}

}

res += '\n';

}

MessageBox.Show(res, "Матрица инцидентности");

}

private void adjMatrixShow\_Click(object sender, EventArgs e)

{

var newM = Main.CreateAdjMatrix();

string res = "";

//res += Main.dots.Count.ToString() + " " + Main.lines.Count.ToString() + "\n";

for (int i = 0; i < Main.dots.Count; i++)

{

for (int j = 0; j < Main.dots.Count; j++)

{

if (j != Main.dots.Count - 1)

{

res += newM[i, j] + " ";

}

else

{

res += newM[i, j];

}

}

res += '\n';

}

MessageBox.Show(res, "Матрица смежности");

}

private void aviMatrixShow\_Click(object sender, EventArgs e)

{

var newM = Main.CreateAviMatrix();

string res = "";

for (int i = 0; i < Main.dots.Count; i++)

{

for (int j = 0; j < Main.dots.Count; j++)

{

if (j != Main.dots.Count - 1)

{

res += newM[i, j] + " ";

}

else

{

res += newM[i, j];

}

}

res += '\n';

}

MessageBox.Show(res, "Матрица достижимости");

}

}

}

Содержимое файла Main.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace GrafBuilder

{

public static class Main

{

public static int dotCount = 0;

public static int dotMax = 10;

public static int lineCount = 0;

public static int activeDotInd = 0;

public static int mouseX;

public static int mouseY;

public static string sourcePath = "";

public static string destPath = "";

public static List<Dot> dots = new List<Dot>();

public static List<Line> lines = new List<Line>();

public static void CreateDots(int n)

{

int r = 100;

int OX = 250;

int OY = 200;

for(int i = 1; i <= n; i++)

{

double x, y;

x = OX + r \* Math.Cos(2 \* i \* Math.PI / n);

y = OY + r \* Math.Sin(2 \* i \* Math.PI / n);

dotCount++;

dots.Add(new Dot((int)x, (int)y, dotCount));

}

}

public static bool CheckLineAviability(Dot firstDot, Dot secondDot)

{

foreach(Line x in lines)

{

if (x.dot1 == firstDot && x.dot2 == secondDot)

return false;

}

return true;

}

public static Dot FindByInd(int ind)

{

Dot temp = new Dot(0, 0, 0);

for (int i = 0; i < dots.Count(); i++)

{

if (dots[i].DotInd == ind)

{

temp = dots[i];

}

}

return temp;

}

public static int[,] CreateIncMatrix()

{

int[,] resultMatrix = new int[dots.Count, lines.Count];

for(int i = 0; i < dots.Count; i++)

{

for(int j = 0; j < lines.Count; j++)

{

resultMatrix[i, j] = 0;

}

}

for(int j = 0; j < lines.Count; j++)

{

resultMatrix[lines[j].dot1.DotInd - 1, j] = 1;

resultMatrix[lines[j].dot2.DotInd - 1, j] = -1;

}

return resultMatrix;

}

public static int[,] CreateAdjMatrix()

{

int[,] res = new int[dots.Count, dots.Count];

for(int i = 0; i < dots.Count; i++)

{

for(int j = 0; j < dots.Count; j++)

{

res[i, j] = 0;

}

}

foreach(var x in lines)

{

res[x.dot1.DotInd - 1, x.dot2.DotInd - 1] = 1;

}

return res;

}

public static int[,] Multiplication(int[,] a, int[,] b)

{

int[,] r = new int[a.GetLength(0), b.GetLength(1)];

for (int i = 0; i < a.GetLength(0); i++)

{

for (int j = 0; j < b.GetLength(1); j++)

{

for (int k = 0; k < b.GetLength(0); k++)

{

r[i, j] += a[i, k] \* b[k, j];

}

}

}

return r;

}

public static int[,] CreateAviMatrix()

{

int[,] res = CreateAdjMatrix();

var adjMatrix = CreateAdjMatrix();

var firstOperand = adjMatrix;

for (int n = 1; n < dots.Count; n++)

{

int[,] tempRes = Multiplication(firstOperand, adjMatrix);

for (int i = 0; i < dots.Count; i++)

{

for (int j = 0; j < dots.Count; j++)

{

res[i, j] += tempRes[i, j];

}

}

firstOperand = tempRes;

}

return res;

}

}

}

Содержимое файла Dot.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace GrafBuilder

{

public class Dot

{

public Dot()

{

DotX = 0;

DotY = 0;

}

public Dot(int x, int y)

{

DotX = x;

DotY = y;

}

public Dot(int x, int y, int n)

{

DotX = x;

DotY = y;

DotInd = n;

}

public Dot ConvertToDec()

{

Dot res = new Dot();

res.dotX = DotX;

res.dotY = 500 - DotY;

return res;

}

public Dot ConvertToParent()

{

Dot res = new Dot();

res.dotX = DotX;

res.dotY = 500 - DotY;

return res;

}

public Dot Substraction(Dot dot1, Dot dot2)

{

return new Dot(dot1.DotX - dot2.DotX, dot1.DotY - dot2.DotY);

}

public Dot Addiction(Dot dot1, Dot dot2)

{

return new Dot(dot1.DotX + dot2.DotX, dot1.DotY + dot2.DotY);

}

public Dot Rotation(Dot dot, double angle)

{

Dot dotCopy = dot;

angle = angle \* Math.PI / 180;

dotCopy.dotX = (int)(dot.DotX \* Math.Cos(angle) - dot.DotY \* Math.Sin(angle));

dotCopy.dotY = (int)(dot.DotX \* Math.Sin(angle) + dot.dotY \* Math.Cos(angle));

return dotCopy;

}

public string ShowCoords()

{

return DotX.ToString() + DotY.ToString();

}

private int dotInd;

public int DotInd

{

get { return dotInd; }

set { dotInd = value; }

}

private int dotX;

public int DotX

{

get { return dotX; }

set { dotX = value; }

}

private int dotY;

public int DotY

{

get { return dotY; }

set { dotY = value; }

}

}

}

Содержимое файла Line.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace GrafBuilder

{

public class Line

{

public Dot dot1;

public Dot dot2;

public int line\_ind;

private int arrowLength = 30;

private double arrowAngle = 30;

public Dot firstArrow = new Dot();

public Dot secondArrow = new Dot();

public void UpdateLine()

{

Dot B = dot1.ConvertToDec();

Dot A = dot2.ConvertToDec();

int fullLength = (int)Math.Sqrt(Math.Pow(B.DotX - A.DotX, 2) + Math.Pow(B.DotY - A.DotY, 2));

firstArrow.DotX = (int)(A.DotX + (B.DotX - A.DotX) \* ((double)arrowLength / (double)fullLength));

firstArrow.DotY = (int)(A.DotY + (B.DotY - A.DotY) \* ((double)arrowLength / (double)fullLength));

//firstArrow = firstArrow.ConvertToParent();

secondArrow = firstArrow;

firstArrow = firstArrow.Substraction(firstArrow, A);

secondArrow = secondArrow.Substraction(secondArrow, A);

firstArrow = firstArrow.Rotation(firstArrow, arrowAngle);

secondArrow = secondArrow.Rotation(secondArrow, -arrowAngle);

firstArrow = firstArrow.Addiction(firstArrow, A);

secondArrow = secondArrow.Addiction(secondArrow, A);

firstArrow = firstArrow.ConvertToParent();

secondArrow = secondArrow.ConvertToParent();

}

public Line(Dot first\_dot, Dot second\_dot, int ind)

{

dot1 = first\_dot;

dot2 = second\_dot;

line\_ind = ind;

UpdateLine();

}

}

}

# ПРИЛОЖЕНИЕ Б

Функционирование программы



Рисунок Б.1 – Отрисовка импортированного графа.

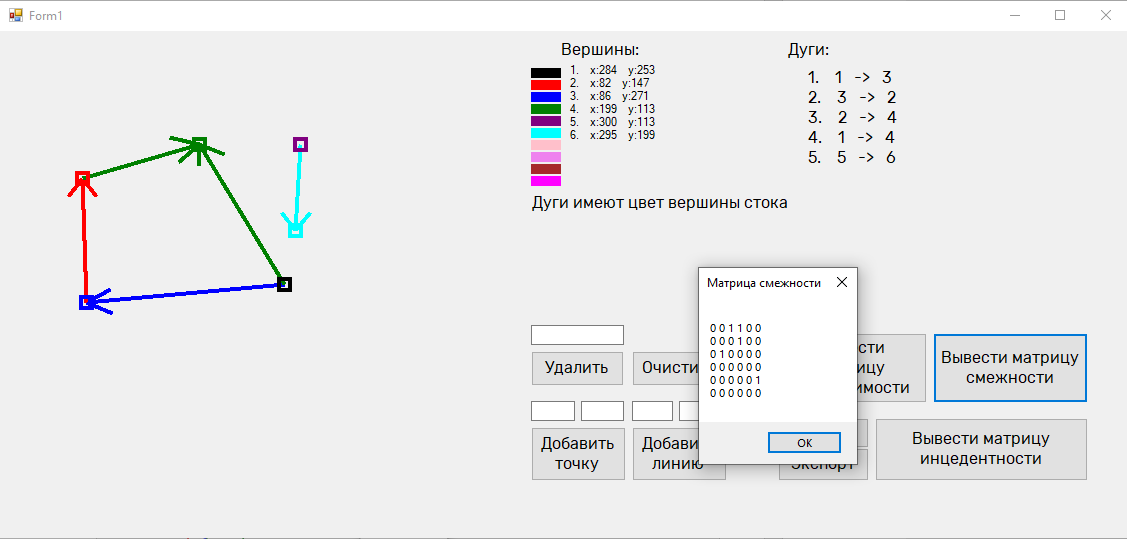


Рисунок Б.2 – Матрица смежности для этого графа

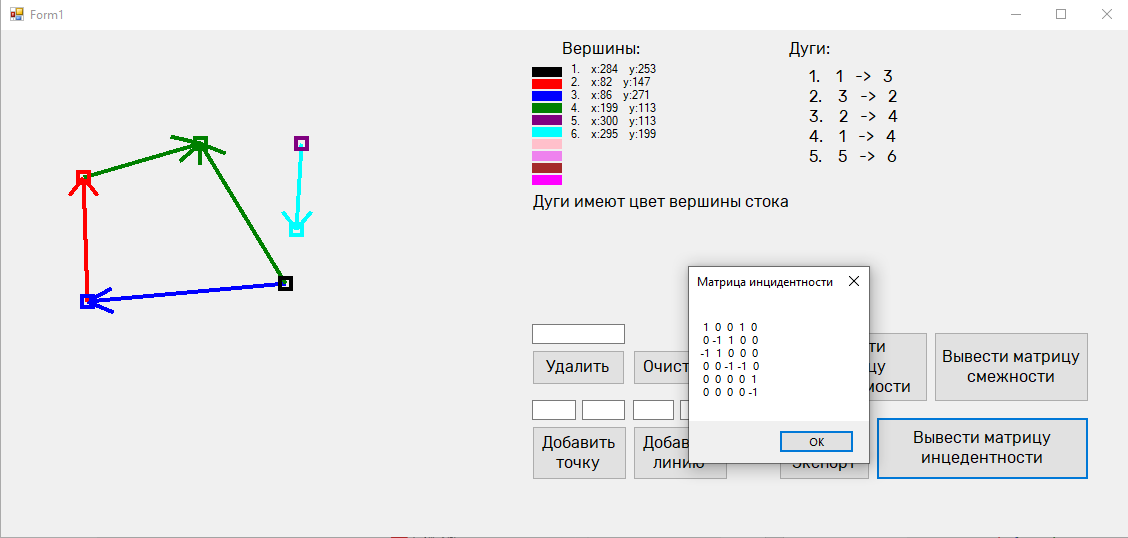


Рисунок Б.3 – Матрица инцидентности для этого графа.

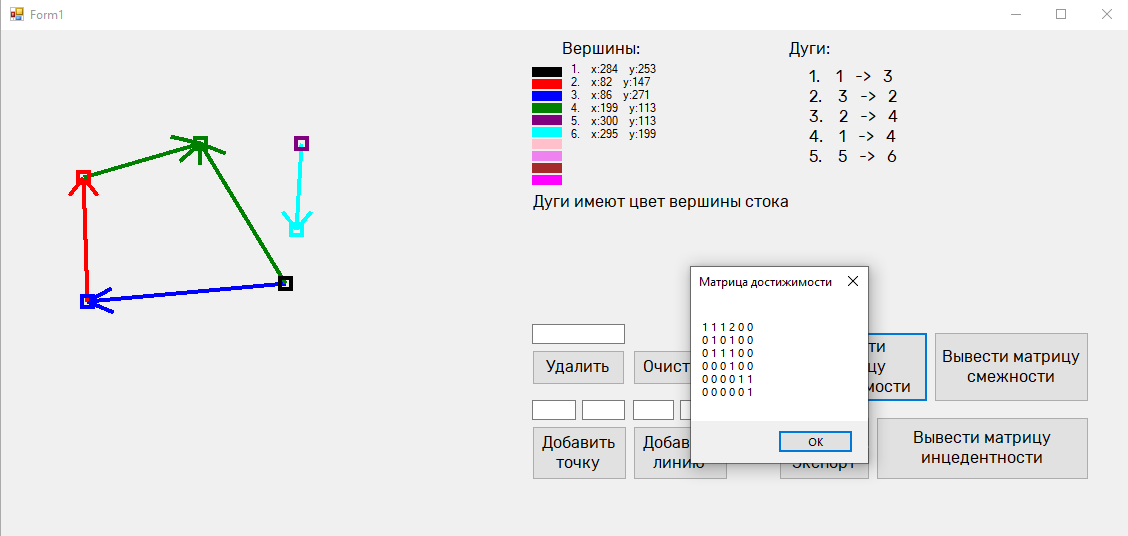


Рисунок Б.4 – Матрица достижимости для этого графа.