Липецкий государственный технический университет

Факультет автоматизации и информатики

Кафедра Автоматизированных систем управления

ЛАБОРАТОРНАЯ РАБОТА №5

По дисциплине «Системный анализ»

Анализ качества структуры на основе структурных характеристик системы

Студент Чаплыгин И.С.

Группа ПИ-18

Руководитель

Профессор Качановский Ю.П.

Липецк 2021 г.

Задание кафедры

Запрограммировать методику расчета группы показателей качества структуры системы. Использовать заданное исходное описание графа. В случае невозможности рассчитать показатель для ориентированного графа-преобразовать граф в неориентированный.

Вариант 9: Дан граф в виде множества правых инциденций G+, получить связности A_{Σ} , С.

Пример работы программы

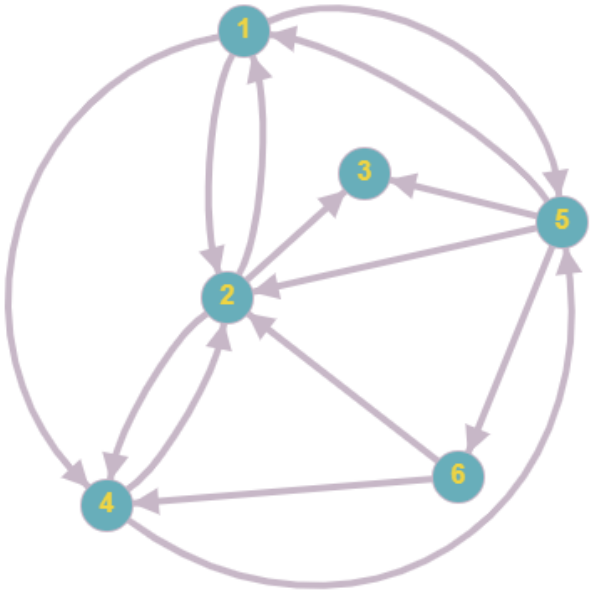
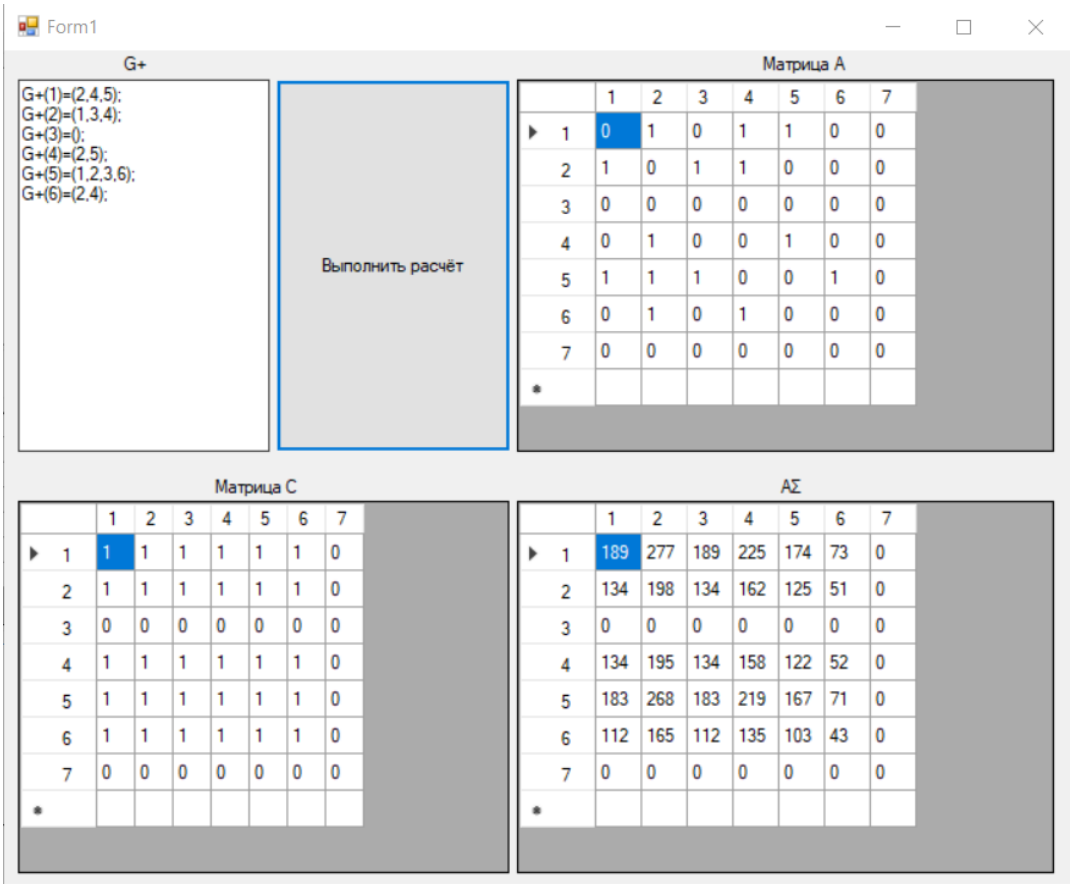


Рисунок 2 – Входной граф



	1	2	3	4	5	6	7
1	0	1	0	1	1	0	0
2	1	0	1	1	0	0	0
3	0	0	0	0	0	0	0
4	0	1	0	0	1	0	0
5	1	1	1	0	0	1	0
6	0	1	0	1	0	0	0
7	0	0	0	0	0	0	0

	1	2	3	4	5	6	7
1	1	1	1	1	1	1	0
2	1	1	1	1	1	1	0
3	0	0	0	0	0	0	0
4	1	1	1	1	1	1	0
5	1	1	1	1	1	1	0
6	1	1	1	1	1	1	0
7	0	0	0	0	0	0	0

	1	2	3	4	5	6	7
1	189	277	189	225	174	73	0
2	134	198	134	162	125	51	0
3	0	0	0	0	0	0	0
4	134	195	134	158	122	52	0
5	183	268	183	219	167	71	0
6	112	165	112	135	103	43	0
7	0	0	0	0	0	0	0

Рисунок 3 – Результат работы программы

Код программы

```
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace Lab5
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }


        void toMxAfromGP(ref int[,] matrix, ref int row)
        {
            StringBuilder str = new StringBuilder(IncidencePlus.Text);
            if (str.Length < 9)
            {
                return;
            }
            for (int i = 0; i < str.Length; ++i)
            {
                if (str[i] == '\n' || str[i] == ' ' || str[i] == '\r')
                {
                    str.Remove(i, 1);
                    --i;
                }
            }
            while (str[str.Length - 1] == ';')
            {
                str.Remove(str.Length - 1, 1);
            }
            string strb = str.ToString();
            string[] list = strb.Split(new char[] { ';' },
StringSplitOptions.RemoveEmptyEntries);

            int rows = list.Count<string>();
            row = rows;
            matrix = new int[rows, rows];
            for (int i = 0; i < rows; ++i)
            {
                for (int j = 0; j < rows; ++j)
                {
                    matrix[i, j] = 0;
                }
            }

            for (int i = 0; i < list.Length; ++i)
            {
                list[i] = list[i].Replace("G+(", "");
                list[i] = list[i].Replace(")=(", ",");
                list[i] = list[i].Replace(")", "");
                string[] list2 = list[i].Split(new char[] { ',' },
StringSplitOptions.RemoveEmptyEntries);
                try
                {
                    int numrow = Convert.ToInt32(list2[0]) - 1;
                    Convert.ToInt32(list2[1]);
                    for (int j = 1; j < list2.Length; ++j)
```

```
            {
                int numcolumn = Convert.ToInt32(list2[j]) - 1;
                matrix[numrow, numcolumn] = 1;
            }
        }
        catch
        {

        }
    }
}

void nodeAnalysis(int[,,] mxASum, int rows, ref int[,] mxA)
{
    mxASum = new int[rows, rows, rows];

    for (int i = 0; i < rows; i++)
    {
        for (int j = 0; j < rows; j++)
        {
            mxASum[0, i, j] = mxA[i, j];
        }
    }

    for (int k = 1; k < rows; k++)
    {
        for (int i = 0; i < rows; i++)
        {
            for (int j = 0; j < rows; j++)
            {
                mxASum[k, i, j] = 0;
                for (int m = 0; m < rows; m++)
                {
                    mxASum[k, i, j] += mxASum[(k - 1), i, m] * mxA[m, j];
                }
            }
        }
    }

    int[,] newMatrix = new int[rows, rows];
    for (int i = 0; i < rows; i++)
    {
        for (int j = 0; j < rows; ++j)
        {
            newMatrix[i, j] = 0;
            for (int k = 0; k < rows; ++k)
            {
                newMatrix[i, j] += mxASum[k, i, j];
            }
        }
    }
    mxASum = new int[rows, rows, 1];
    for (int i = 0; i < rows; i++)
    {
        for (int j = 0; j < rows; j++)
        {
            mxASum[i, j, 0] = newMatrix[i, j];
            Console.Write(mxASum[i, j, 0] + " ");
        }
        Console.WriteLine();
    }
    Console.WriteLine();
    for (int i = 0; i < rows; i++)
    {
        for (int j = 0; j < rows; j++)
```

```
            {
                if (newMatrix[i, j] >= 1)
                {
                    newMatrix[i, j] = 1;
                }
                Console.Write(newMatrix[i, j] + " ");
            }
            Console.WriteLine();
        }
        this.fromMxAIntsToTbA(ref mxASum, rows);
        this.fromMxAIntsToTbC(ref newMatrix, rows);
    }

    private void fromMxAIntsToTbA(ref int[,,] mx, int rows)
    {
        this.MatrixA.Rows.Clear();
        this.MatrixA.Columns.Clear();
        for (int i = 0; i < rows; ++i)
        {
            this.MatrixA.Columns.Add("", (i + 1).ToString());
            this.MatrixA.Columns[i].Width = 30;
            this.MatrixA.Rows.Add();
            this.MatrixA.Rows[i].HeaderCell.Value = (i + 1).ToString();
        }
        for (int i = 0; i < rows; ++i)
        {
            for (int j = 0; j < rows; ++j)
            {
                this.MatrixA.Rows[i].Cells[j].Value = mx[i, j, 0].ToString();
            }
        }
    }
    private void fromMxAIntsToTbA2(ref int[,] mx, int rows)
    {
        this.MatrixA2.Rows.Clear();
        this.MatrixA2.Columns.Clear();
        for (int i = 0; i < rows; ++i)
        {

            MatrixA2.Columns.Add("", (i + 1).ToString());
            MatrixA2.Columns[i].Width = 30;
            MatrixA2.Rows.Add();
            MatrixA2.Rows[i].HeaderCell.Value = (i + 1).ToString();
        }
        for (int i = 0; i < rows; ++i)
        {
            for (int j = 0; j < rows; ++j)
            {
                this.MatrixA2.Rows[i].Cells[j].Value = mx[i, j].ToString();
            }
        }
    }
    private void fromMxAIntsToTbC(ref int[,] mx, int rows)
    {
        this.MatrixC.Rows.Clear();
        this.MatrixC.Columns.Clear();
        for (int i = 0; i < rows; ++i)
        {

            this.MatrixC.Columns.Add("", (i + 1).ToString());
            this.MatrixC.Columns[i].Width = 25;
            this.MatrixC.Rows.Add();
            this.MatrixC.Rows[i].HeaderCell.Value = (i + 1).ToString();
        }
        for (int i = 0; i < rows; ++i)
```

```
            {
                for (int j = 0; j < rows; ++j)
                {
                    this.MatrixC.Rows[i].Cells[j].Value = mx[i, j].ToString();
                }
            }
        }

        int getLine(ref int[,] mx, int rows, int vFinish, List<int> nodes)
        {
            if (nodes[nodes.Count - 1] == vFinish)
            {
                int sum = 0;
                for (int i = 0; i < nodes.Count - 1; ++i)
                {
                    if (mx[nodes[i], nodes[i + 1]] != 0)
                    {
                        sum += mx[nodes[i], nodes[i + 1]];
                    }
                }
                return sum;
            }
            else
            {
                int sum = 0;
                for (int i = 0; i < rows; ++i)
                {
                    if (nodes.IndexOf(i) < 0)
                    {
                        List<int> tmpList = new List<int>();
                        tmpList = nodes;
                        tmpList.Add(i);
                        sum += getLine(ref mx, rows, vFinish, tmpList);
                    }
                }
                return sum;
            }
        }
        private void Form1_Load(object sender, EventArgs e)
        {
        }

        private void button1_Click_1(object sender, EventArgs e)
        {
            this.MatrixA.Columns.Clear();
            this.MatrixA.Rows.Clear();
            int rows = 0;
            int[,] mx = null;
            this.toMxAfromGP(ref mx, ref rows);
            this.fromMxAIntsToTbA2(ref mx, rows);
            this.nodeAnalysis(null, rows, ref mx);
        }
    }
}
```