**ГОСУДАРСТВЕННОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ**

**ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ**

**"ДОНЕЦКИЙ НАЦИОНАЛЬНЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ"**

Отчет по индивидуальному заданию по курсу «КДМ»

Тема: «Поиск компонент связности»

|  |  |
| --- | --- |
| Руководитель:  кафедры ПИ | Выполнил:  студент гр. ПИ-18б  Моргунов А. Г. |

Донецк – 2019

ВВЕДЕНИЕ

Дискре́тная матема́тика  — часть математики, изучающая [дискретные](https://ru.wikipedia.org/wiki/%D0%94%D0%B8%D1%81%D0%BA%D1%80%D0%B5%D1%82%D0%BD%D0%BE%D1%81%D1%82%D1%8C) [математические структуры](https://ru.wikipedia.org/wiki/%D0%9C%D0%B0%D1%82%D0%B5%D0%BC%D0%B0%D1%82%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B0%D1%8F_%D1%81%D1%82%D1%80%D1%83%D0%BA%D1%82%D1%83%D1%80%D0%B0), такие, как [графы](https://ru.wikipedia.org/wiki/%D0%93%D1%80%D0%B0%D1%84_(%D0%BC%D0%B0%D1%82%D0%B5%D0%BC%D0%B0%D1%82%D0%B8%D0%BA%D0%B0)) и [утверждения в логике](https://ru.wikipedia.org/wiki/%D0%92%D1%8B%D1%81%D0%BA%D0%B0%D0%B7%D1%8B%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5_(%D0%BB%D0%BE%D0%B3%D0%B8%D0%BA%D0%B0))[1].

В контексте [математики](https://ru.wikipedia.org/wiki/%D0%9C%D0%B0%D1%82%D0%B5%D0%BC%D0%B0%D1%82%D0%B8%D0%BA%D0%B0) в целом дискретная математика часто отождествляется с конечной математикой — направлением, изучающим конечные структуры — [конечные графы](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BD%D0%B5%D1%87%D0%BD%D1%8B%D0%B9_%D0%B3%D1%80%D0%B0%D1%84), [конечные группы](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BD%D0%B5%D1%87%D0%BD%D0%B0%D1%8F_%D0%B3%D1%80%D1%83%D0%BF%D0%BF%D0%B0), [конечные автоматы](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BD%D0%B5%D1%87%D0%BD%D1%8B%D0%B5_%D0%B0%D0%B2%D1%82%D0%BE%D0%BC%D0%B0%D1%82%D1%8B). И при этом можно выделить некоторые особенности, не присущие разделам, работающим с бесконечными и непрерывными структурами. Так, в дискретных направлениях как правило обширнее класс разрешимых задач, так как во многих случаях возможен [полный перебор](https://ru.wikipedia.org/wiki/%D0%9F%D0%BE%D0%BB%D0%BD%D1%8B%D0%B9_%D0%BF%D0%B5%D1%80%D0%B5%D0%B1%D0%BE%D1%80) вариантов, тогда как в разделах, имеющих дело с бесконечными и непрерывными структурами, для разрешимости обычно требуются существенные ограничения на условия. В этой же связи в дискретной математике особо важную роль играют задачи построения конкретных [алгоритмов](https://ru.wikipedia.org/wiki/%D0%90%D0%BB%D0%B3%D0%BE%D1%80%D0%B8%D1%82%D0%BC), и в том числе, эффективных с точки зрения [вычислительной сложности](https://ru.wikipedia.org/wiki/%D0%92%D1%8B%D1%87%D0%B8%D1%81%D0%BB%D0%B8%D1%82%D0%B5%D0%BB%D1%8C%D0%BD%D0%B0%D1%8F_%D1%81%D0%BB%D0%BE%D0%B6%D0%BD%D0%BE%D1%81%D1%82%D1%8C). Ещё одна особенность дискретной математики — невозможность применения для её экстремальных задач техник [анализа](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B0%D0%BB%D0%B8%D0%B7_(%D1%80%D0%B0%D0%B7%D0%B4%D0%B5%D0%BB_%D0%BC%D0%B0%D1%82%D0%B5%D0%BC%D0%B0%D1%82%D0%B8%D0%BA%D0%B8)), существенно использующих недоступные для дискретных структур понятия [гладкости](https://ru.wikipedia.org/wiki/%D0%93%D0%BB%D0%B0%D0%B4%D0%BA%D0%B0%D1%8F_%D1%84%D1%83%D0%BD%D0%BA%D1%86%D0%B8%D1%8F). В широком смысле, дискретной математикой могут считаться охваченными значительные части [алгебры](https://ru.wikipedia.org/wiki/%D0%90%D0%BB%D0%B3%D0%B5%D0%B1%D1%80%D0%B0), [теории чисел](https://ru.wikipedia.org/wiki/%D0%A2%D0%B5%D0%BE%D1%80%D0%B8%D1%8F_%D1%87%D0%B8%D1%81%D0%B5%D0%BB), [математической логики](https://ru.wikipedia.org/wiki/%D0%9C%D0%B0%D1%82%D0%B5%D0%BC%D0%B0%D1%82%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B0%D1%8F_%D0%BB%D0%BE%D0%B3%D0%B8%D0%BA%D0%B0).

В рамках учебных программ дискретная математика обычно рассматривается как совокупность разделов, связанных с приложениями к [информатике](https://ru.wikipedia.org/wiki/%D0%98%D0%BD%D1%84%D0%BE%D1%80%D0%BC%D0%B0%D1%82%D0%B8%D0%BA%D0%B0) и [вычислительной технике](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D0%BF%D1%8C%D1%8E%D1%82%D0%B5%D1%80): [теория функциональных систем](https://ru.wikipedia.org/wiki/%D0%A2%D0%B5%D0%BE%D1%80%D0%B8%D1%8F_%D1%84%D1%83%D0%BD%D0%BA%D1%86%D0%B8%D0%BE%D0%BD%D0%B0%D0%BB%D1%8C%D0%BD%D1%8B%D1%85_%D1%81%D0%B8%D1%81%D1%82%D0%B5%D0%BC_(%D0%B4%D0%B8%D1%81%D0%BA%D1%80%D0%B5%D1%82%D0%BD%D0%B0%D1%8F_%D0%BC%D0%B0%D1%82%D0%B5%D0%BC%D0%B0%D1%82%D0%B8%D0%BA%D0%B0)), [теория графов](https://ru.wikipedia.org/wiki/%D0%A2%D0%B5%D0%BE%D1%80%D0%B8%D1%8F_%D0%B3%D1%80%D0%B0%D1%84%D0%BE%D0%B2), [теория автоматов](https://ru.wikipedia.org/wiki/%D0%A2%D0%B5%D0%BE%D1%80%D0%B8%D1%8F_%D0%B0%D0%B2%D1%82%D0%BE%D0%BC%D0%B0%D1%82%D0%BE%D0%B2), [теория кодирования](https://ru.wikipedia.org/wiki/%D0%A2%D0%B5%D0%BE%D1%80%D0%B8%D1%8F_%D0%BA%D0%BE%D0%B4%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F), [комбинаторика](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D0%B1%D0%B8%D0%BD%D0%B0%D1%82%D0%BE%D1%80%D0%B8%D0%BA%D0%B0), [целочисленное программирование](https://ru.wikipedia.org/wiki/%D0%A6%D0%B5%D0%BB%D0%BE%D1%87%D0%B8%D1%81%D0%BB%D0%B5%D0%BD%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5).

1 ОПИСАНИЕ АЛГОРИТМА НАХОЖДЕНИЯ ЧИСЛА КОМПОНЕНТ СВЯЗНОСТИ

Исходныме данные. На вход подается матрица смежности, определяющая граф. В ячейке первого столбца первой строки пишется количество вершин.

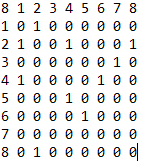
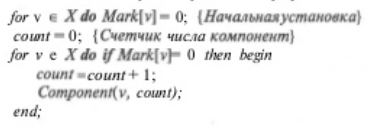


Рисунок 1.1 – Пример входных данных

Текст алгоритма из книжки[2](Рисунок 1.2)



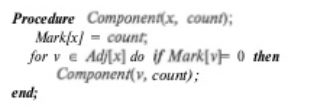


Рисунок 1.2 – Текст алгоритма из книжки

Алгоритм идет по связным вершинам, попутно отмечая их, пока может и заканчивает работу с отдельной компонентой связности, когда больше некуда идти. Затем он приступает к обработке следующей компоненты связности, пока вершины в графе не закончатся.

При работе возникла необходимость модифицировать алгоритм, для обеспечения вывода листинга программы в файл. Добавлен дополнительный аргумент в функцию Component(поток вывода в файл).

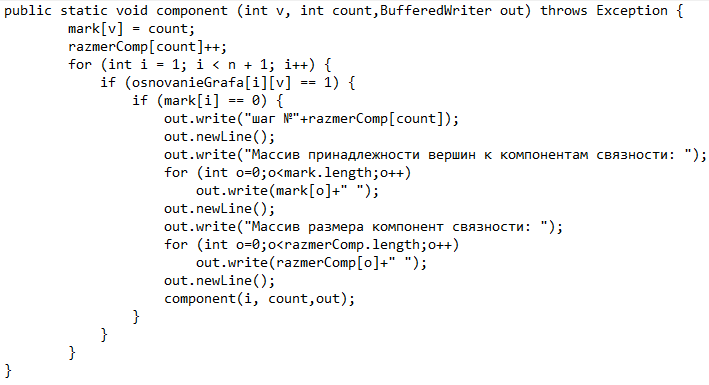


Рисунок 1.3 – Текст алгоритма реализованный в программе

2 КОНТРОЛЬНЫЕ ПРИМЕРЫ

2.1 Контрольный пример, решенный вручную

Исходные данные

6 1 2 3 4 5 6

1 0 1 1 1 0 0

2 1 0 0 0 0 0

3 1 0 0 1 1 0

4 1 0 1 0 0 0

5 0 0 1 0 0 0

6 0 0 0 0 0 0

массив принадлежности компонент изначально имеет вид 0 0 0 0 0 0 0. Элемент с индексом 0 не задействуется, а в дальнейшем индексы в массиве соответствуют номеру вершины, к которой они относятся.

Массив размера компонента связности, так же как и массив принадлежности компонент не задействует элемент с индексом 0, а при использовании в программе индексы массива соответствуют номерам компонент связности. Этот массив изначально имеет вид 0 0 0 0 0 0 0

1я компонента связности

Берем вершину 1, отмечаем ее как пройденную(массив принадлежности компонент переходит из вида 0 0 0 0 0 0 0 к виду 0 1 0 0 0 0 0) и переходим ко 2й вершине, отмечаем ее(массив принадлежности компонент переходит из вида 0 1 0 0 0 0 0 к виду 0 1 1 0 0 0 0), переходим к 3й, отмечаем ее(массив принадлежности компонент переходит из вида 0 1 1 0 0 0 0 к виду 0 1 1 1 0 0 0), переходим к 4й, отмечаем ее(массив принадлежности компонент переходит из вида 0 1 1 1 0 0 0 к виду 0 1 1 1 1 0 0), переходим к 5й(т.к. она связана с 3й, а 4 и 3 уже отмечены как пройденные) (массив принадлежности компонент переходит из вида 0 1 1 1 1 0 0 к виду 0 1 1 1 1 1 0),. Работа с 1й компонентой связности заканчивается т.к больше нет вершин в которые можно попасть из уже пройденных вершин. Каждый раз когда отмечается вершина элемент массива размера компонент связности (с индексом равным номеру компоненты связности) увеличивается на 1. После работы с 1й компонентой связности этот массив имеет вид 0 5 0 0 0 0 0 (т.к. было отмечено 5 вершин при работе с 1й компонентой связности)

2я компонента связности

Берем 6ю вершину, отмечаем ее(массив принадлежности компонент переходит из вида 0 1 1 1 1 1 0 к виду 0 1 1 1 1 1 2),. Нет вершин связанных с 6й, значит заканчиваем работу со 2й компонентой связности. После работы с 2й компонентой связности массив размера компонент связности имеет вид 0 5 1 0 0 0 0 (т.к. было отмечено 1 вершин при работе со 2й компонентой связности)

Алгоритм заканчивает свою работу т.к. больше нет неотмеченных вершин.

Результат контрольного примера: 2 компоненты связности. 1я включающая в себя вершины 1, 2, 3, 4, 5 и 2я включающая в себя вершину 6.

2.2 Листинг контрольного примера в программе

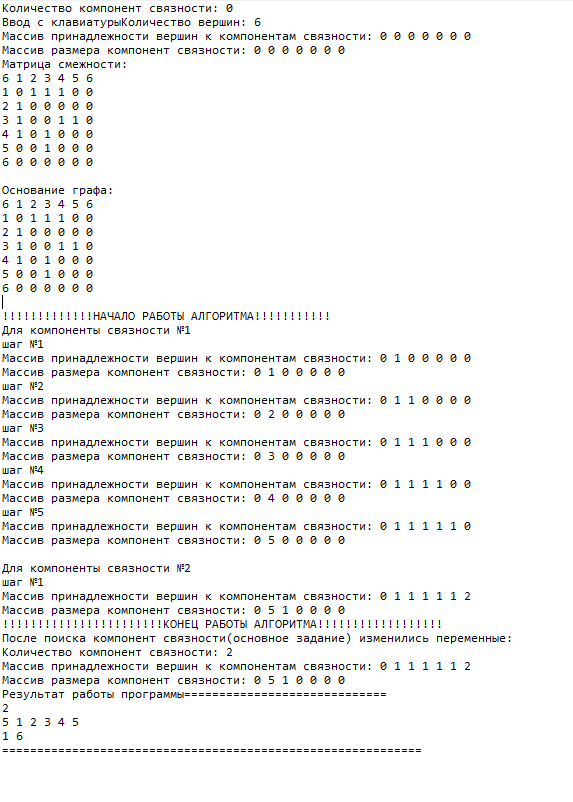


Рисунок 2.1 – Листинг контрольного примера

2.3 Дополнительные контрольные примеры

2.3.1 Двудольный граф

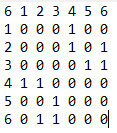


Рисунок 2.2 – Исходные данные для двудольного графа

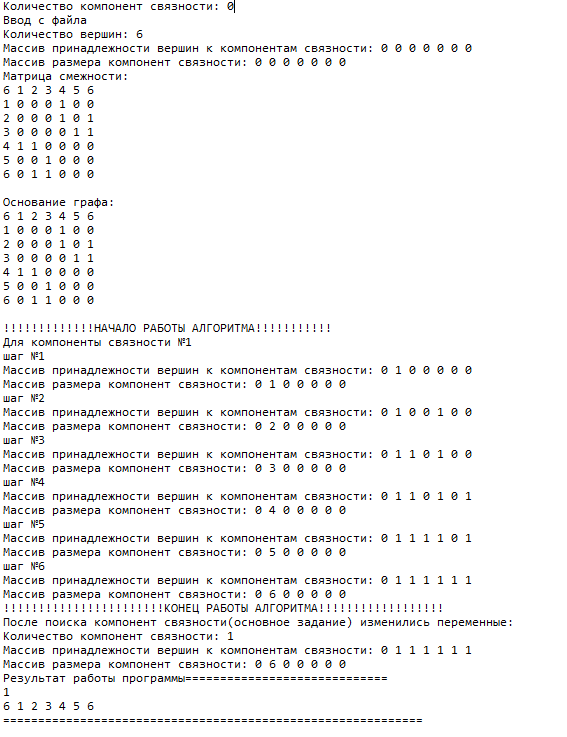


Рисунок 2.3 – Листинг для двудольного графа

2.3.2 Клика

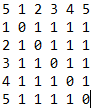


Рисунок 2.4 – Исходные данные для клики

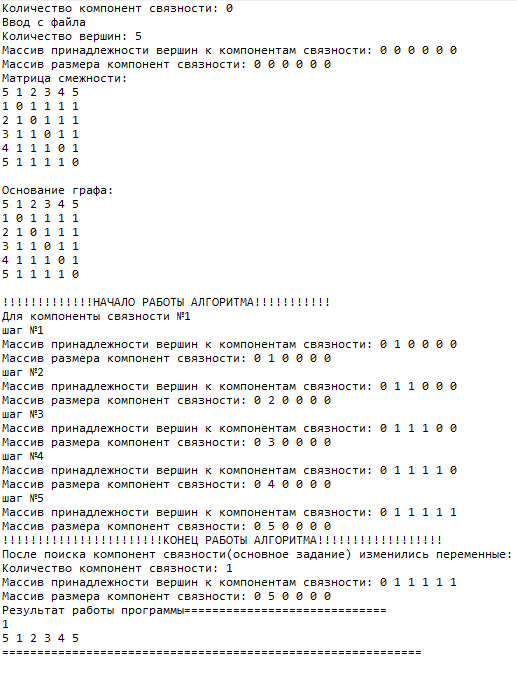


Рисунок 2.5 – Листинг для клики

2.3.3 Несвязный

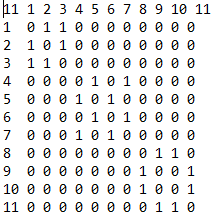


Рисунок 2.6 – Исходные данные для несвязного графа

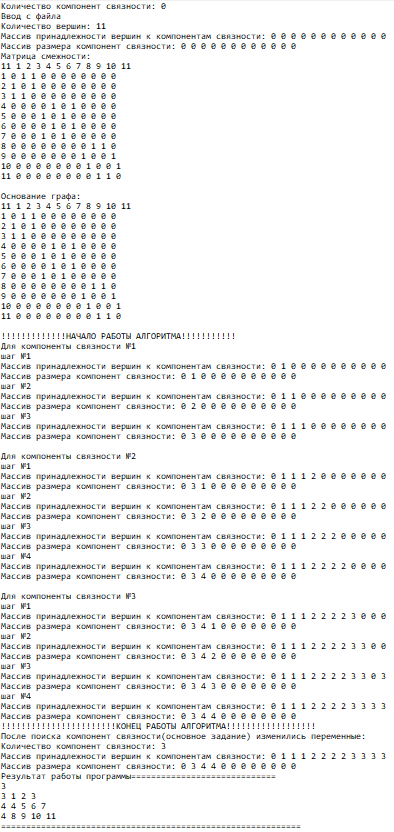


Рисунок 2.7 – Листинг для несвязного графа

2.3.4 Орграф



Рисунок 2.8 – Исходные данные для орграфа

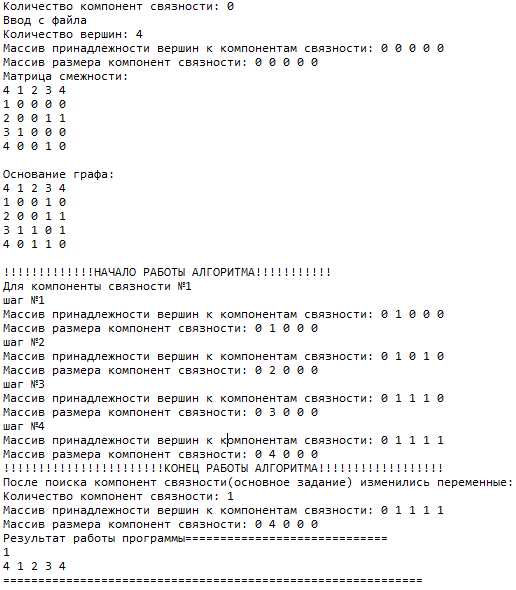


Рисунок 2.9 – Листинг для орграфа

2.3.5 Полный двудольный

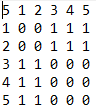


Рисунок 2.10 – Исходные данные для полного двудольного графа

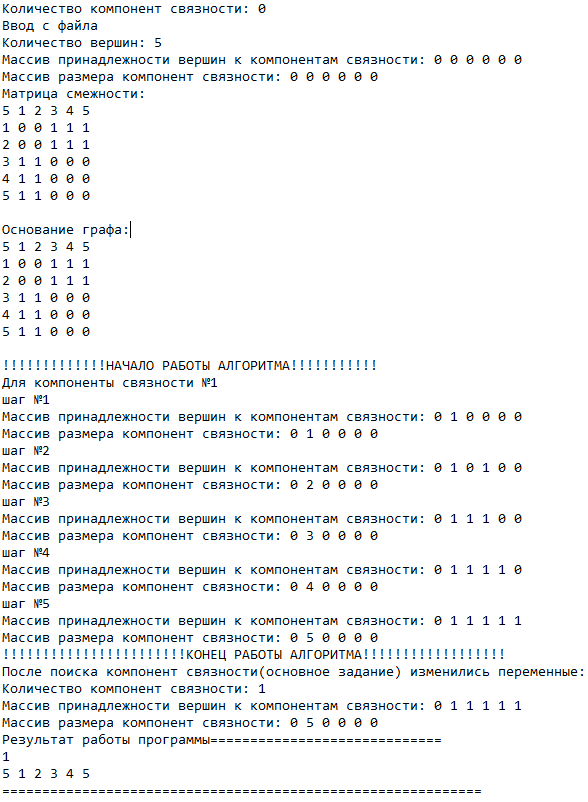


Рисунок 2.11 – Листинг для полного двудольного графа

2.3.6 Простая цепь

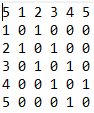


Рисунок 2.12 – Исходные данные для простой цепи

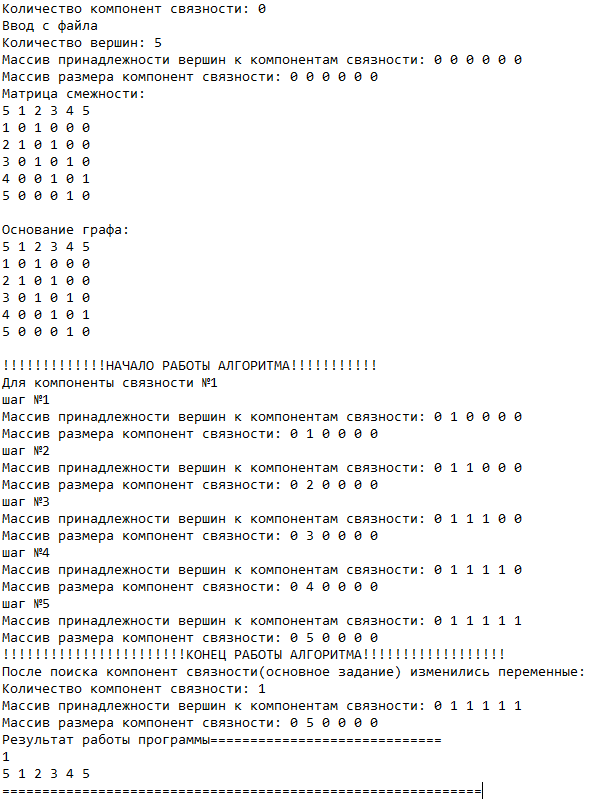


Рисунок 2.13 – Листинг для простой цепи

2.3.7 Простой цикл

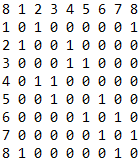


Рисунок 2.14 – Исходные данные для простого цикла

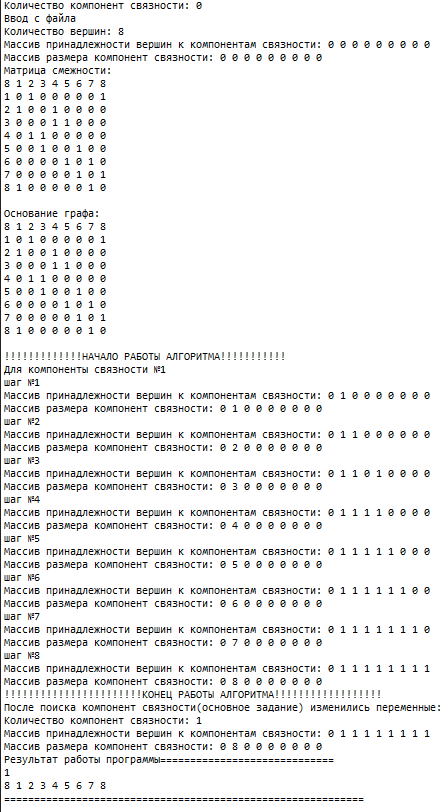


Рисунок 2.15 – Листинг для простого цикла

2.3.8 Тривиальный



Рисунок 2.16 – Исходные данные для тривиального графа

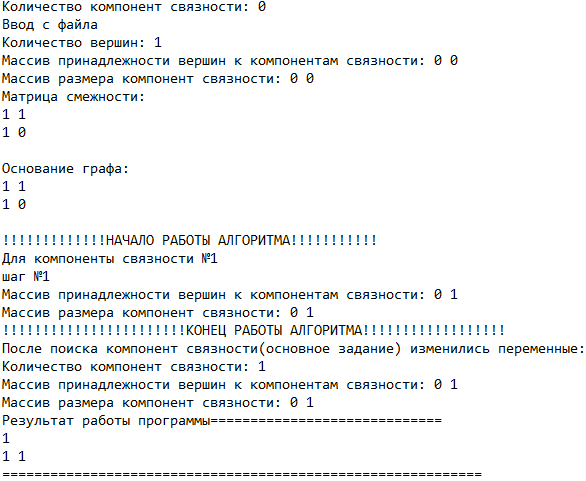


Рисунок 2.17 – Листинг для тривиального графа

2.3.9 Звезда

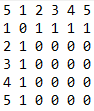


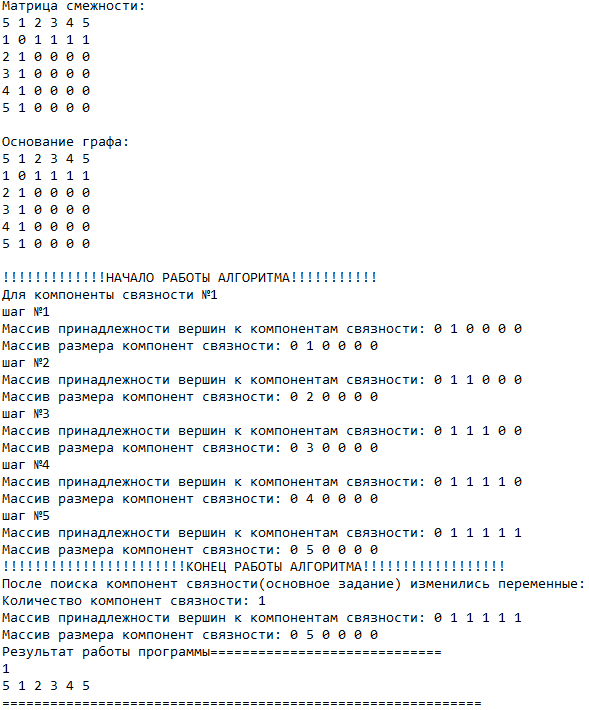
Рисунок 2.18 – Исходные данные для звезды

Рисунок 2.19 – Листинг для звезды

2.3.10 Односторонний орграф



Рисунок 2.20 – Исходные данные для одностороннего орграфа

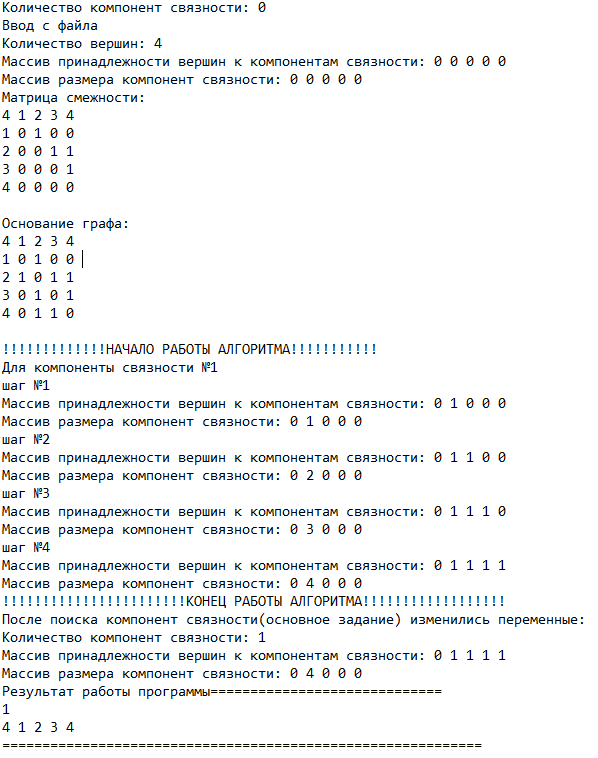


Рисунок 2.21 – Листинг для одностороннего орграфа

2.3.11 Сильный орграф



Рисунок 2.22 – Исходные данные для сильного орграфа

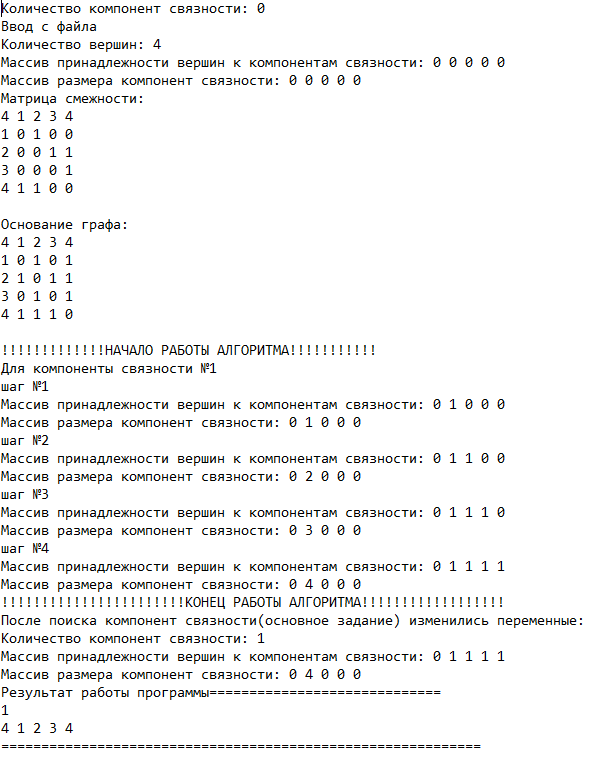


Рисунок 2.23 – Листинг для сильного орграфа

2.3.12 Слабый орграф



Рисунок 2.22 – Исходные данные для слабого орграфа

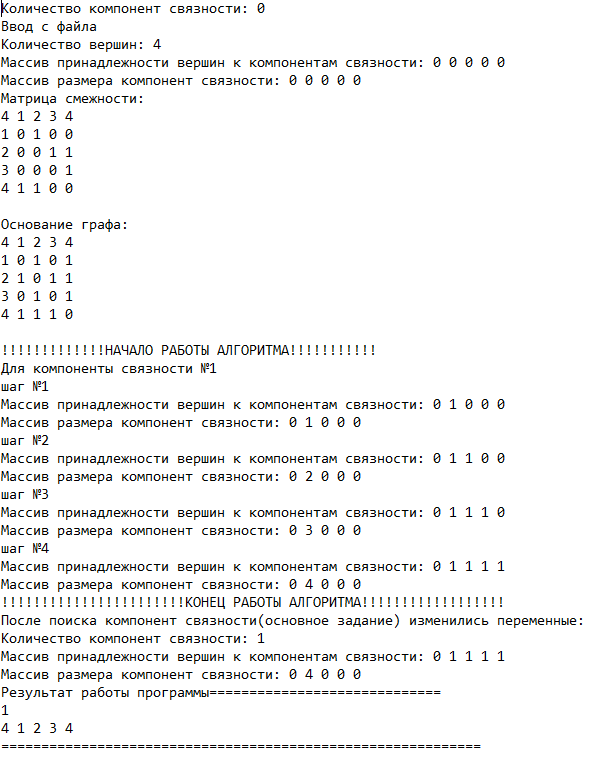


Рисунок 2.23 – Листинг для слабого орграфа

2.3.13 Пустой граф



Рисунок 2.24 – Исходные данные для пустого графа

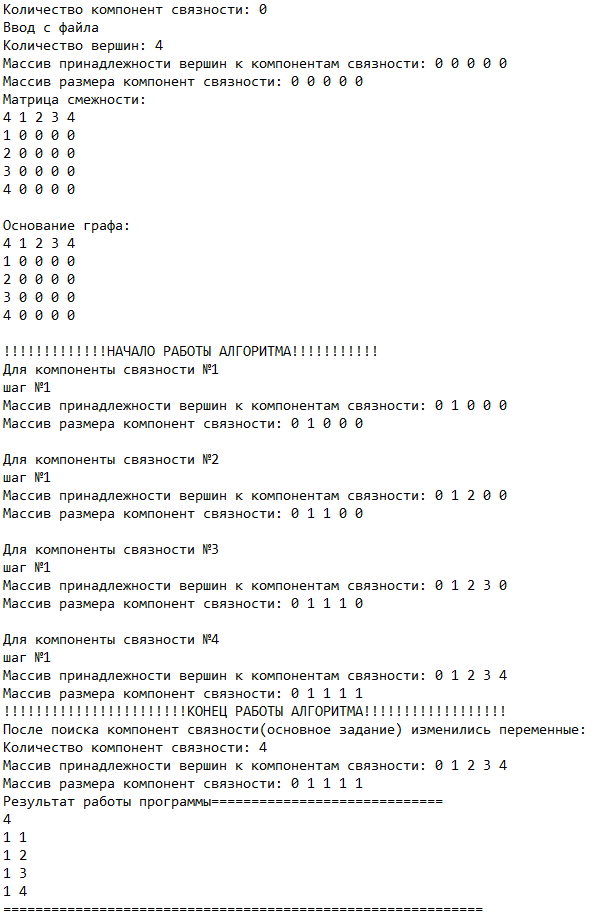


Рисунок 2.25 – Листинг для пустого графа

3 ОПИСАНИЕ ПРОГРАММЫ

Программа ищет количество компонент связности, количество и номера вершин, состоящих в каждой компоненте. При работе с неографами используется таблица смежности, при работе с орграфами используется основание данного графа. Входные данные можно подавать как с клавиатуры так и из файла. Результат работы программы выводится динамически в окно программы, а также записывается в файл листинга программы. Предусмотрены и по максимуму предотвращены все возможные механические ошибки со стороны пользователя(например матрица смежности при вводе с клавиатуры сразу появляется заполненой нулями, а не пустой, а также главная диагональ и строка и столбец с номерами вершин являются нередактируемыми)

ВЫВОДЫ

В результате мы имеем программу, которая может определять количество компонент связности для неографов и орграфов. Данная программа обладает такими преимуществами как простота в использовании, интуитивно понятный интерфейс, возможность визуализации графов, защищенность от случайных ошибок со стороны пользователя.

К недостаткам программы относятся трудность использования для пользователей, не обладающих базовой теорией относительно графов, неудобство при работе с большим количеством вершин.

Алгоритм поиска компонент связности используется для решения задачи анализа структурной отказоустойчивости вычислительной системы. С его помощью вводится и описывается понятие d-ограниченной компоненты связности(d-компоненты связности), выделяющей в графе G(V, E) максимальный связный подграф Gd′(Vd′,Ed′) с диаметром, не превышающим предельного значения d: ∀u, v∈Vd′ d(u, v) ≤ d. Очевидно, что при d(G′) ≤ d d-компонента связности Gd′(Vd′, Ed′) совпадает с компонентой G′(V′, E′) и Gd′(Vd′, Ed′) ≡ G′(V ′, E′), а при d(G′) > d она является ее подграфом Gd′(Vd′, Ed′)∈G′(V ′, E′).[3]

4 ИНСТРУКЦИИ ПОЛЬЗОВАТЕЛЮ

Для запуска программы необходимо произвести запуск испоняемого файла KDM\_individual\_task.jar.

Формат файла с исходными данными(текстовый файл):

Первый символ - количество вершин в графе

Остальные данные – матрица смежности для графа с заданным количеством вершин.

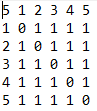


Рисунок 4.1 – Исходные данные

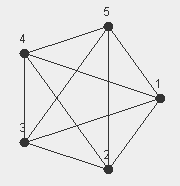


Рисунок 4.2 – Граф построенный по матрице смежности(Рис 4.1)

Файл листинга имеет следующую структуру

1й блок-значение переменных до начала программы

2й блок – матрица смежности и основание графа

3й блок – работа алгоритма

4й блок – значение переменных после работы программы

5й блок – результат работы программы

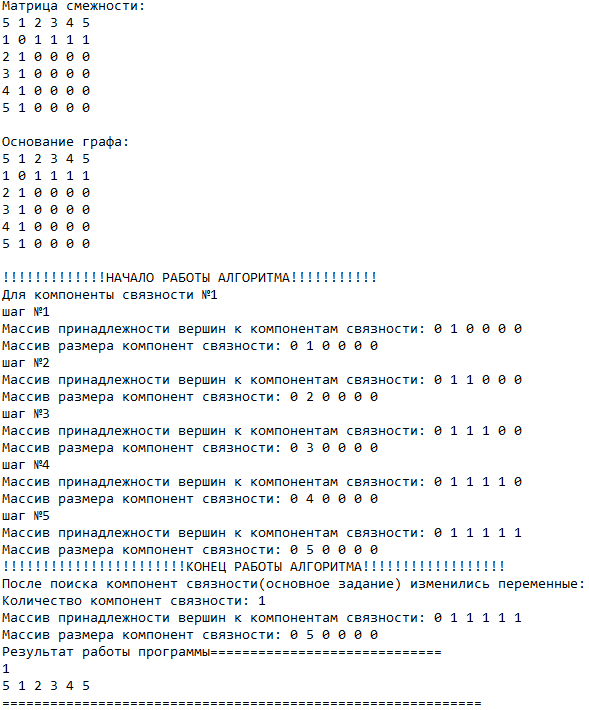


Рисунок 4.3 – Пример листинга программы

Для работы пользователя с программой предоствалсяется несколько сценариев:

1. Работа с файлами. Пользователь должен выбрать файл при помощи кнопки «выбрать файл» и выбрать с каким графом (орграфом или неографом) он хочет работать и нажать на кнопку «Пуск!»
2. Ввод с клавиатуры. Пользователь должен задать размер матрицы, построить ее с помощью кнопки «Задать размер матрицы», выбрать пункт ввод вручную, заполнить построенную матрицу и нажать на «Пуск!»

Результаты:

1е число – количество компонент связности

Числа в последующих строках это количество вершин в данной компоненте связности и номера этих вершин по возрастанию

5 ЛИСТИНГ С ПОДРОБНЫМ ОПИСАНИЕМ

Файл Main.java--------------------------------------------------------------------------

import javax.swing.\*;

import javax.swing.filechooser.FileNameExtensionFilter;

import javax.swing.table.DefaultTableColumnModel;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.io.\*;

import java.util.Arrays;

import java.util.InputMismatchException;

import java.util.NoSuchElementException;

import java.util.Scanner;

public class Main {

public final static double PI = 3.14159265359;

public static int[][] graf = null;//{{5,1,2,3,4,5},{1,1,0,0,0,0},{2,0,1,0,0,0},{3,0,0,1,0,0},{4,0,0,0,1,0},{5,0,0,0,0,1}};//матрица смежности

public static int[][] osnovanieGrafa;

public static int count = 0, n;//колво компонент связности, колво вершин

public static int[] mark = null;//{0,0,0,0,0,0};

public static int[] razmerComp;

public static File file= new File("In.txt");

public static int orientirovannost1=0;

public static void main(String[] args) throws Exception {

JFrame frame = new JFrame();//главное окно

frame.setSize(1000, 1000);

frame.setTitle("Individual task");

frame.setLayout(null);

JRadioButton neoRadioButton = new JRadioButton("Неограф", true);

neoRadioButton.setBounds(400, 600, 200, 25);

JRadioButton orRadioButton = new JRadioButton("Орграф", false);

orRadioButton.setBounds(400, 625, 200, 25);

ButtonGroup orientirovannost = new ButtonGroup();

orientirovannost.add(neoRadioButton);

orientirovannost.add(orRadioButton);

JTextArea listing = new JTextArea();

JScrollPane listingScroll = new JScrollPane(listing);

listingScroll.setBounds(600,600,380,350);

JRadioButton fileRadioButton = new JRadioButton("Ввод с файла", true);

fileRadioButton.setBounds(400, 650, 200, 25);

JRadioButton handsRadioButton = new JRadioButton("Ввод вручную", false);

handsRadioButton.setBounds(400, 700, 200, 25);

ButtonGroup vvod = new ButtonGroup();

vvod.add(handsRadioButton);

vvod.add(fileRadioButton);

JTextField tableRazmer=new JTextField();

tableRazmer.setBounds(400,725,200,25);

JButton chooseFile = new JButton();

chooseFile.setText("Выбрать файл");

chooseFile.setBounds(400,675,200,25);

chooseFile.addActionListener(new ActionListener() {//выбор файла

@Override

public void actionPerformed(ActionEvent e) {

JFileChooser fileChooser = new JFileChooser();

FileNameExtensionFilter filter = new FileNameExtensionFilter("Текстовый файл","txt");//ограничение на текстовые файлы

fileChooser.setFileFilter(filter);

fileChooser.setFileSelectionMode(JFileChooser.FILES\_ONLY);

int ret = fileChooser.showDialog(null,"Open file");

if(ret==JFileChooser.APPROVE\_OPTION){

file = fileChooser.getSelectedFile();

}

}

});

GraphicsPanel graphicsPanel = new GraphicsPanel();//создание компонента, в котором мы будем строить граф

final JScrollPane scroll = new JScrollPane(graphicsPanel);

scroll.setBounds(0, 0, 980, 600);

MyTableModel tm = new MyTableModel();

DefaultTableColumnModel cm = new DefaultTableColumnModel();

JTable table = new JTable(tm);

table.setColumnModel(cm);

table.setTableHeader(null);

table.setAutoResizeMode(0);

JButton paintButton = new JButton();

paintButton.setText("Пуск!");

paintButton.setSize(100, 100);

paintButton.setBounds(400, 780, 200, 50);

paintButton.addActionListener(new ActionListener() {//активация алгоритма

@Override

public void actionPerformed(ActionEvent e) {

try (BufferedWriter out = new BufferedWriter(new FileWriter("Listing.txt"))) {//создание потока вывода в файл

count = 0;//задаем стартовое количество компонент связности

out.write("Количество компонент связности: "+count);

if (orRadioButton.isSelected())

orientirovannost1 = 1;//если выбран орграф то задаем значение флага 1

else

orientirovannost1 = 0;//если выбран неограф то задаем значение флага 0

if (fileRadioButton.isSelected()) {//выбран вывод из файла

out.newLine();

out.write("Ввод с файла");

out.newLine();

try (BufferedReader in = new BufferedReader(new FileReader(file))) {//vvod s fayla

Scanner sin = new Scanner(in);

n = sin.nextInt();//присваиваем количеству вершин первое число из файла

graf = new int[n + 1][n + 1];

graf[0][0] = n;

mark = new int[n + 1];

razmerComp = new int[n + 1];

for (int i = 1; i < n + 1; i++)//заполняем массив матрицы смежности

graf[0][i] = sin.nextInt();//заполняем массив матрицы смежности

for (int i = 1; i < n + 1; i++)//заполняем массив матрицы смежности

for (int j = 0; j < n + 1; j++) {//заполняем массив матрицы смежности

graf[i][j] = sin.nextInt();//заполняем массив матрицы смежности

}

} catch (IOException | NumberFormatException e1) {

JOptionPane.showMessageDialog(null, "File Error");

out.newLine();

out.write("File Error");

out.newLine();

return;

} catch (InputMismatchException e1) {

JOptionPane.showMessageDialog(null, "Ошибка в содержании файла");

out.newLine();

out.write("Ошибка в содержании файла");

out.newLine();

return;

} catch (NoSuchElementException e1) {

JOptionPane.showMessageDialog(null, "Недостаточно данных в файле(неполная таблица смежности)");

out.newLine();

out.write("Недостаточно данных в файле(неполная таблица смежности)");

out.newLine();

return;

}

}//if

if (handsRadioButton.isSelected()) {//если выбран ввод с клавиатуры

out.newLine();

out.write("Ввод с клавиатуры");

try {

n = (Integer) (table.getValueAt(0, 0));//сохраняем количество

} catch (ArrayIndexOutOfBoundsException e1) {

JOptionPane.showMessageDialog(null, "Сначала создайте и заполните матрицу смежности");

out.newLine();

out.write("Сначала создайте и заполните матрицу смежности");

out.newLine();

return;

}

graf = new int[n + 1][n + 1];

mark = new int[n + 1];

razmerComp = new int[n + 1];

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

for (int j = 0; j <= n; j++) {

try {

graf[i][j] = Integer.parseInt(table.getValueAt(i, j).toString());//заполнение массива матрицы смежности

} catch (NumberFormatException e1) {

JOptionPane.showMessageDialog(null, "Недопустимое значение (строка: " + i + " столбец: " + j + ")");

out.newLine();

out.write("Недопустимое значение (строка: " + i + " столбец: " + j + ")");

out.newLine();

return;

}

if (i > 0 & j > 0)

if (graf[i][j] != 0 & graf[i][j] != 1) {

JOptionPane.showMessageDialog(null, "Недопустимое значение (строка: " + i + " столбец: " + j + ")");

out.newLine();

out.write("Недопустимое значение (строка: " + i + " столбец: " + j + ")");

out.newLine();

return;

}

}

}

for (int i = 1; i < n + 1; i++) {//zapolnenie massiva komponent

mark[i] = 0;//задаем начальные значение массивов

razmerComp[i] = 0;//задаем начальные значение массивов

}

System.out.println("Граф");

printMatr(graf);

System.out.println("=============================================");

System.out.println("Основание графа");

osnovanieGrafa = normalize(graf);//поиск основания для графа

printMatr(osnovanieGrafa);

out.write("Количество вершин: "+n);

out.newLine();

out.write("Массив принадлежности вершин к компонентам связности: ");

for (int i=0;i<mark.length;i++)

out.write(mark[i]+" ");

out.newLine();

out.write("Массив размера компонент связности: ");

for (int i=0;i<razmerComp.length;i++)

out.write(razmerComp[i]+" ");

out.newLine();

out.write("Матрица смежности:\n");

for(int i =0;i<n+1;i++) {

for (int j = 0; j < n + 1; j++) {

out.write(Integer.toString(graf[i][j])+ " ");

}

out.newLine();

}

out.newLine();

out.write("Основание графа:\n");

for(int i =0;i<n+1;i++) {

for (int j = 0; j < n + 1; j++) {

out.write(Integer.toString(osnovanieGrafa[i][j])+ " ");

}

out.newLine();

}

out.newLine();

if (orientirovannost1 == 0) {//если пользователь выбрал неограф, а исходные данные задают орграф то программа автоматически берет основание данного орграфа

int a = 0;//флаг

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

for (int j = 1; j < n + 1; j++) {

if (graf[i][j] != graf[j][i])//если матрица смежности несимметрична

a++;//увеличиваем значение флага

}

if (a != 0) {//если флаг не 0

JOptionPane.showMessageDialog(null, "Матрица смежности определяет орграф поэтому для работы программы было взято основание введенного орграфа");

out.write("Матрица смежности определяет орграф поэтому для работы программы было взято основание введенного орграфа");

out.newLine();

}

}

out.write("!!!!!!!!!!!!!НАЧАЛО РАБОТЫ АЛГОРИТМА!!!!!!!!!!!");

for (int i = 1; i < n + 1; i++) {//poisk komponent svyaznosti

if (mark[i] == 0) {//если вершина i еще не помечена

count++;//увеличиваем число компонент связности

out.newLine();

out.write("Для компоненты связности №"+count);

out.newLine();

component(i, count,out);//запускаем функцию работающую с компонентой связности

out.write("шаг №"+razmerComp[count]);

out.newLine();

out.write("Массив принадлежности вершин к компонентам связности: ");

for (int o=0;o<mark.length;o++)

out.write(mark[o]+" ");

out.newLine();

out.write("Массив размера компонент связности: ");

for (int o=0;o<razmerComp.length;o++)

out.write(razmerComp[o]+" ");

out.newLine();

}

}

out.write("!!!!!!!!!!!!!!!!!!!!!!!КОНЕЦ РАБОТЫ АЛГОРИТМА!!!!!!!!!!!!!!!!!!");

out.newLine();

out.write("После поиска компонент связности(основное задание) изменились переменные:");

out.newLine();

out.write("Количество компонент связности: "+count);

out.newLine();

out.write("Массив принадлежности вершин к компонентам связности: ");

for (int i=0;i<mark.length;i++)

out.write(mark[i]+" ");

out.newLine();

out.write("Массив размера компонент связности: ");

for (int i=0;i<razmerComp.length;i++)

out.write(razmerComp[i]+" ");

out.newLine();

System.out.println(count);

String exit = new String();

out.write("Результат работы программы=============================\n" + Integer.toString(count));

exit += "Компонент связности в графе: " + count + "\nКоличество вершин в компоненте : номера вершин\n";

out.newLine();

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

if (razmerComp[i] != 0) {

out.write(Integer.toString(razmerComp[i]) + " ");

exit += (razmerComp[i]) + " : ";

}

for (int j = 1; j < n + 1; j++) {

if (mark[j] == i) {

out.write(Integer.toString(graf[j][0]) + "; ");

exit += (graf[j][0]) + "; ";

}

}

if (razmerComp[i] != 0)

exit += "\n";

if (razmerComp[i] != 0)

out.newLine();

}

listing.setText(exit);

out.write("============================================================");

out.flush();

} catch (IOException ex) {

JOptionPane.showMessageDialog(null,"Ошибка при выводе в файл");

return;

} catch (Exception ex) {

ex.printStackTrace();

}

graphicsPanel.n = n;

graphicsPanel.repaint();

}

});

JButton setRazmer = new JButton();

setRazmer.setText("Задать размер матрицы");

setRazmer.setBounds(400,750,200,25);

setRazmer.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

try {

if(Integer.parseInt(tableRazmer.getText())<1||Integer.parseInt(tableRazmer.getText())>101)

{

JOptionPane.showMessageDialog(null,"Введите число от 1 до 100");

}else {

tm.setColumnCount(Integer.parseInt(tableRazmer.getText()) + 1);

tm.setRowCount(Integer.parseInt(tableRazmer.getText()) + 1);

table.setValueAt(Integer.parseInt(tableRazmer.getText()), 0, 0);

for (int i = 1; i <= Integer.parseInt(tableRazmer.getText()); i++) {

table.setValueAt(i, i, 0);

}

for (int i = 1; i <= Integer.parseInt(tableRazmer.getText()); i++) {

table.setValueAt(i, 0, i);

}

for (int i = 1; i <= Integer.parseInt(tableRazmer.getText()); i++)

for (int j = 1; j <= Integer.parseInt(tableRazmer.getText()); j++)

table.setValueAt(0, i, j);

for (int i =0;i<Integer.parseInt(tableRazmer.getText())+1;i++){

table.getColumnModel().getColumn(i).setMaxWidth(25);

}

}

}catch (Exception e3){

JOptionPane.showMessageDialog(null,"Введите число");

}

}

});

JScrollPane scrollTable = new JScrollPane(table);

scrollTable.setBounds(0, 600, 400, 350);

frame.add(listingScroll);

frame.add(chooseFile);

frame.add(setRazmer);

frame.add(tableRazmer);

frame.add(fileRadioButton);

frame.add(handsRadioButton);

frame.add(neoRadioButton);

frame.add(orRadioButton);

frame.add(paintButton);

frame.add(scroll);

frame.add(scrollTable);

frame.setLocationRelativeTo(null);

frame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

frame.setVisible(true);

}//main

public static void component (int v, int count,BufferedWriter out) throws Exception {//функция поиска составляющих в компоненте связности аргументы:номер вершины, количество компонент связноси, поток вывода в файл

mark[v] = count;//прсваиваем элементу массива с номером обрабатываемой вершины значение равному номеру обрабатываемой компоненты связности

razmerComp[count]++;//увеличиваем значение элемента массива с номером обрабатываемой компоненты связности

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

if (osnovanieGrafa[i][v] == 1) {//если текущая вершина связана с проверяемой

if (mark[i] == 0) {//если вершина еще не обработана

out.write("шаг №"+razmerComp[count]);

out.newLine();

out.write("Массив принадлежности вершин к компонентам связности: ");

for (int o=0;o<mark.length;o++)

out.write(mark[o]+" ");

out.newLine();

out.write("Массив размера компонент связности: ");

for (int o=0;o<razmerComp.length;o++)

out.write(razmerComp[o]+" ");

out.newLine();

component(i, count,out);//рекурсивно вызываем функцию обработки

}

}

}

}//component

public static void printMatr(int[][] a) {//вывод матрицы в консоль

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

for (int j = 0; j < a.length; j++) {

System.out.print(a[i][j] + " ");

}

System.out.println();

}

}//printmatr

public static int[][] normalize(int[][] in) {//поиск основания матрицы

int[][] out = new int[n + 1][n + 1];

for (int i = 0; i < n + 1; i++)

out[i] = Arrays.copyOf(graf[i], n + 1);

for (int i = 1; i < in.length; i++)

for (int j = 1; j < in.length; j++) {

if (in[i][j] == 1 | in[j][i] == 1) {

out[i][j] = 1;

out[j][i] = 1;

}

}

return out;

}//normalize

}

Файл GraphicPanel,java-----------------------------------------------------------

import javax.swing.\*;

import java.awt.\*;

import java.lang.Exception;

public class GraphicsPanel extends JPanel {

public static int n = Main.n;

@Override

public void paintComponent(Graphics g) {

Graphics2D g2 = (Graphics2D) g;

super.paintComponent(g2);

int i = 1;

int a = 0;

int[] indOfEl;

int o;

if (n > 0) {

try {

while (i <= n && Main.razmerComp[i] != 0) {

o = 0;

indOfEl = new int[Main.razmerComp[i]];

for (int y = 1; y < n + 1; y++) {

if (Main.mark[y] == i & o <= indOfEl.length) {

indOfEl[o] = y;

o++;

}

}

addGraph(g2, Main.razmerComp[i], 30, 30, a, indOfEl);

a += Main.razmerComp[i];

i++;

repaint();

}

} catch (Exception e) {

e.printStackTrace();

}

} else {

//JOptionPane.showMessageDialog(null, "Введенный граф не имеет вершин");

return;

}//if

setPreferredSize(new Dimension(40 \* (n + 3), 40 \* (n + 3)));

setSize(40 \* (n + 3), 40 \* (n + 3));

}

//paintComponent

public static void drawGraph(Graphics2D g, int n, double xhl, double yhl, int[] indOfEl) throws Exception {//рисование графа

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

g.fillOval((int) ((xhl + n \* 30 / 2) + (n \* 30 / 2) \* (Math.cos(Math.toRadians((360 / n) \* i)))) - 5, (int) ((yhl + n \* 30 / 2) + (n \* 30 / 2) \* (Math.sin(Math.toRadians((360 / n) \* i)))) - 5, 10, 10);

g.drawString(Integer.toString(indOfEl[i]), (int) ((xhl + n \* 30 / 2) + (n \* 30 / 2) \* (Math.cos(Math.toRadians((360 / n) \* i))) - 5), (int) ((yhl + n \* 30 / 2) + (n \* 30 / 2) \* (Math.sin(Math.toRadians((360 / n) \* i)))) - 10);

for (int j = 0; j < n; j++) {

if (Main.graf[indOfEl[i]][indOfEl[j]] == 1 & indOfEl[i] != indOfEl[j]) {

if(Main.orientirovannost1==0)

g.drawLine((int) ((xhl + n \* 30 / 2) + (n \* 30 / 2) \* (Math.cos(Math.toRadians((360 / n) \* i)))), (int) ((yhl + n \* 30 / 2) + (n \* 30 / 2) \* (Math.sin(Math.toRadians((360 / n) \* i)))), (int) ((xhl + n \* 30 / 2) + (n \* 30 / 2) \* (Math.cos(Math.toRadians((360 / n) \* j)))), (int) ((yhl + n \* 30 / 2) + (n \* 30 / 2) \* (Math.sin(Math.toRadians((360 / n) \* j)))));

else{

drawArrowLine(g,(int) ((xhl + n \* 30 / 2) + (n \* 30 / 2) \* (Math.cos(Math.toRadians((360 / n) \* i)))), (int) ((yhl + n \* 30 / 2) + (n \* 30 / 2) \* (Math.sin(Math.toRadians((360 / n) \* i)))), (int) ((xhl + n \* 30 / 2) + (n \* 30 / 2) \* (Math.cos(Math.toRadians((360 / n) \* j)))), (int) ((yhl + n \* 30 / 2) + (n \* 30 / 2) \* (Math.sin(Math.toRadians((360 / n) \* j)))),10,3);

// drawArrow(g,(int) ((xhl + n \* 30 / 2) + (n \* 30 / 2) \* (Math.cos(Math.toRadians((360 / n) \* i)))), (int) ((yhl + n \* 30 / 2) + (n \* 30 / 2) \* (Math.sin(Math.toRadians((360 / n) \* i)))), (int) ((xhl + n \* 30 / 2) + (n \* 30 / 2) \* (Math.cos(Math.toRadians((360 / n) \* j)))), (int) ((yhl + n \* 30 / 2) + (n \* 30 / 2) \* (Math.sin(Math.toRadians((360 / n) \* j)))));

}

}

}

}

}//drawGraph

public static void drawArrowLine(Graphics2D g, int x1, int y1, int x2, int y2, int d, int h) {//рисование стрелки

int dx = x2 - x1, dy = y2 - y1;

double D = Math.sqrt(dx\*dx + dy\*dy);

double xm = D - d, xn = xm, ym = h, yn = -h, x;

double sin = dy / D, cos = dx / D;

x = xm\*cos - ym\*sin + x1;

ym = xm\*sin + ym\*cos + y1;

xm = x;

x = xn\*cos - yn\*sin + x1;

yn = xn\*sin + yn\*cos + y1;

xn = x;

int[] xpoints = {x2, (int) xm, (int) xn};

int[] ypoints = {y2, (int) ym, (int) yn};

g.drawLine(x1, y1, x2, y2);

g.fillPolygon(xpoints, ypoints, 3);

}

public static void addGraph(Graphics2D g, int n, double xhl, double yhl, int usedCount, int[] indOfEl) throws Exception {//добавление графа

drawGraph(g, n, xhl + usedCount \* 40, yhl, indOfEl);//рисование графа

}

}

Файл MyTableModel.java--------------------------------------------------------------------

import javax.swing.table.DefaultTableModel;

public class MyTableModel extends DefaultTableModel {

@Override

public boolean isCellEditable(int row,int column){//главная диагональ и столбец 0 и строчка 0 в таблице - нередактируемы

if(row==0||column==0||row==column) return false;

else return true;

}

}

6 ЭКРАННЫЕ ФОРМЫ

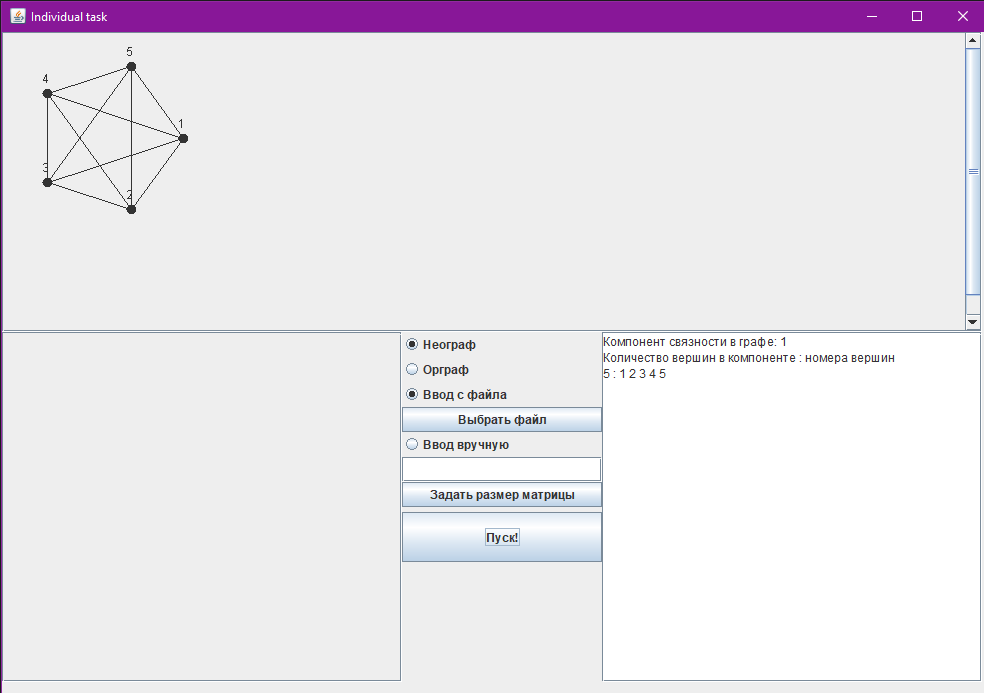


Рисунок 6.1 – Главная форма программы

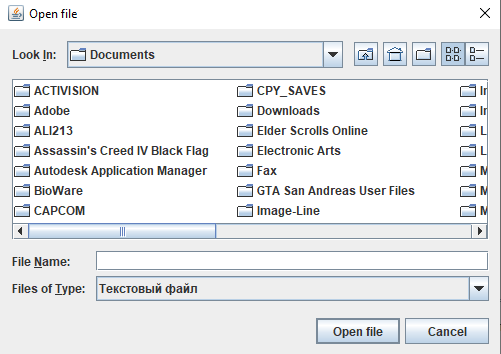


Рисунок 6.2 – Форма выбора файла

СПИСОК ЛИТЕРАТУРЫ

1. Новиков Ф. А. Н73 Дискретная математика дл я программистов: Учебник дл я вузов. 3-е изд. — СПб.: Питер, 2009. — 384 е.: ил. — (Серия «Учебник дл я вузов»),
2. Иванов Б.Н. Дискретная математика. Алгоритмы и программы. Москва. 2003.
3. [Функция](https://cyberleninka.ru/article/n/funktsiya-strukturnoy-otkazoustoychivosti-i-d-ogranichennaya-komponenta-svyaznosti-grafa-vychislitelnoy-sistemy) структурной отказоустойчивости и d-ограниченная компонента связности графа вычислительной системы.