МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МО ЭВМ

ОТЧЕТ

по учебной практике

Тема: Кратчайшие пути в графе. Алгоритм Флойда-Уоршалла

Студентка гр. 7381 _____ Кревчик А.Б.

Студентка гр. 7381 _____ Процветкина А.В.

Студентка гр. 7381 _____ Машина Ю.Д.

Руководитель _____ Размочаева Н.В.

Санкт-Петербург
2019

ЗАДАНИЕ
НА УЧЕБНУЮ ПРАКТИКУ

Студентка Кревчик А.Б. группы 7381

Студентка Процветкина А.В. группы 7381

Студентка Машина Ю.Д. группы 7381

Тема практики: Кратчайшие пути в графе. Алгоритм Флойда-Уоршалла

Задание на практику:

Командная итеративная разработка визуализатора алгоритма(ов) на Java с графическим интерфейсом.

Алгоритм: Флойда-Уоршалла.

Сроки прохождения практики: 01.07.2019 – 14.07.2019

Дата сдачи отчета: 12.07.2019

Дата защиты отчета: 12.07.2019

Студентка	_____	Кревчик А.Б.
Студентка	_____	Процветкина А.В.
Студентка	_____	Машина Ю.Д.
Руководитель	_____	Размочаева Н.В.

АННОТАЦИЯ

Ключевой целью работы является создание GUI-приложения с визуализацией поиска кратчайших путей в графе с помощью алгоритма Флойда-Уоршалла. Графический интерфейс реализован с использованием языка java и платформы JavaFX. Приложение позволяет пользователю увидеть как представление графа, так и каждый шаг алгоритма поиска кратчайших путей в нём. Кроме того, результат работы алгоритма Флойда - Уоршалла при необходимости может быть получен немедленно, а не пошагово. Интерфейс приложения оформлен в приятном для восприятия темном стиле, а специально подобранные шрифты идеально гармонируют с ним. Навигация в приложении осуществляется с помощью интуитивно понятных стильных кнопок. Интефейс содержит элементы pop-up и hover.

SUMMARY

The main goal of this work is to create a GUI-application for finding shortest paths between all pair of vertices in a weighted graph with Floyd-Warshall algorithm. Graphical interface is designed with the help of java language and JavaFX platform. The application allows user to see a representation of the graph as well as every step of the algorithm for finding shortest paths in it. Besides, the result of the Floyd-Warshall algorithm can be shown immediately rather than step-by-step if needed. The interface of the application is designed in an appealing dark style; the fonts chosen amazingly harmonize with it. Navigation in the application works with the help of intuitively clear designed buttons. The interface contains pop-up and hover elements.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

На сегодняшний день знание алгоритмов на графах является важной характеристикой для программиста. Однако представление большинства алгоритмов в форме псевдо-кода плохо способствует их восприятию и усвоению обучающимися. Визуализация же, напротив, позволяет представить даже самый сложный алгоритм естественным и доступным для понимания образом.

Идея текущего проекта состоит визуализации алгоритма Флойда-Уоршалла, позволяющего решить проблему поиска кратчайших путей между всеми парами вершин графа. Работа нацелена на создание понятного графического интерфейса для указанного алгоритма, а также на изучение и закрепление основных возможностей языка программирования Java. Особое внимание уделено оформлению приложения.

1. ТРЕБОВАНИЯ К ПРОГРАММЕ

1.1. Исходные Требования к программе

1.1.1. Требования к вводу исходных данных

Ввод производится из формы GUI, предоставляющей для этого поля́, визуально имитирующие матрицу смежности графа.

1.1.2. Требования к визуализации

У пользователя должна быть возможность проследить за выполнением вычислений пошагово, либо сразу получить результат – матрицу кратчайших путей. Производящиеся на каждом шаге вычисления выводятся в GUI форму. Специально отведенная матрица смежности демонстрирует изменения. Визуализированный граф представлен в комфортном для пользователя виде.

1.2. Уточнение требований после сдачи прототипа

- Добавить имена внизу каждого окна
- Сделать возможным ввод из файла
- Выделять на окне визуализации соответствующие формуле элементы матрицы цветом для удобства восприятия
- Ввод матриц 4-10 вершин
- Визуализация матриц 4-10 вершин
- Кнопка "начать сначала"
- "Шаг назад"

1.3. Уточнение требований после сдачи 1-ой версии

- Диалоговое окно для выбора файла
- Сделать граф с динамикой (в ходе алгоритма граф меняется)

1.4. Уточнение требование после сдачи 2-й версии

- Неизменяемый размер окна

- изменить тип на long для ввода больших чисел
- подсветка ребер и вершин в ходе алгоритма

2. ПЛАН РАЗРАБОТКИ И РАСПРЕДЕЛЕНИЕ РОЛЕЙ В БРИГАДЕ

2.1. План разработки

Основными моментами разработки являются обеспечение корректной работы алгоритма и user-friendly Graphical User Interface (далее GUI).

Разработка разделена на несколько этапов:

1. Набросок GUI, определение его однозначно необходимых базовых возможностей
2. Создание первоначальной версии алгоритма, удовлетворяющей требованиям разработки первоначальной версии GUI
3. Связывание алгоритма с GUI
4. Тестирование первой версии проекта
5. Устранение неполадок
6. Расширение возможностей алгоритма и GUI с целью предоставления бóльшей свободы действий и/или комфорта для пользователя.
7. Последующее тестирование и устранение неполадок

Приоритетом для нас является удобство в использовании нашего проекта, а также обучение и овладение новыми навыками. Бóльшая часть наших сил будет брошена на освоение мастерства разработки GUI и непосредственную работу над ним.

Примерные сроки разработки:

03.07.2019 – согласование набросков GUI и алгоритма.

05.07.2019 – разработанный алгоритм способен выполнять свои базовые функции, к нему создан и привязан базовый интерфейс. Это прототип проекта.

08.07.2019 – исправление неполадок и добавление новых возможностей для пользователя.

10.07.2019 – вторая версия проекта проходит проверку, написана первая версия отчета.

12.07.2019 – финальная версия проекта проходит проверку, написана финальная версия отчета.

2.2. Распределение ролей в бригаде

Согласно предоставленному для ознакомления файлу «технология работы.docx», к участникам бригады применены следующие роли (могут меняться в ходе разработки):

Кревчик А.Б. – разработка основного алгоритма, написание документации,

Процветкина А. В. – разработка тестов и осуществление тестирования, написание документации,

Машина Ю. Д. – проектирование классов, проектирование и разработка GUI.

- Разработчик – студент, участник команды. Ответственность – выполнять текущие задачи итерации, проектировать и реализовывать классы и алгоритмы, загружать код в репозиторий.
- ответственный за итерацию – разработчик, дополнительной обязанностью которого является проверка качества и полноты исполнения задач в трекере и изменение их статуса.

3. ОСОБЕННОСТИ РЕАЛИЗАЦИИ

3.1. Использованные структуры данных

- class **Visualisation**

Атрибуты:

private ResourceBundle resources; // набор ресурсов

private URL location; //Путь до файла, если ввод из файла

private Label if_statement_label; //проверяемое условие i + j < k

private AnchorPane anchorPane; //контейнер отображаемой области

private Label if_statement_result_label; // реальное соотношение между i + j и k

private Button next_step_button; // следующий шаг

private Button immediate_result_button; // получить результат немедленно

private Label k_label; // метка отображающая k в выражении i + j < k

private Label i_label; // метка отображающая i в выражении i + j < k

private Label j_label; // метка отображающая j в выражении i + j < k

private Button previous_step_button; // предыдущий шаг

private Button start_over_button; // начать заново

Методы:

void **initialize**()

Метод, осуществляющий визуальную инициализацию до начала работы алгоритма, т.е. размещение кнопок, привязка обработчиков нажатий, невидимость кнопок вперед и назад, считывание матрицы, объявление элемента подсказки с эффектом hover.

protected void **FloydWarshallAlgo**(int MATRIX_CODE, int [][] dist,Label [][] array_of_labels, int i, int j, int k)

Метод осуществляет шаг вперед алгоритма Флойда-Уоршелла.

protected void **ImmediateFloydWarshallAlgo**(int MATRIXCODE, int [][] dist,Label [][] array_of_labels, int V)

Метод реализует визуализированный переход на последний шаг алгоритма Флойда-Уоршелла.

protected int[][] **floydWarshall**(floydWarshall(int[][] graph, int V)

Метод реализует алгоритм Флойда-Уоршелла без связи с графическим интерфейсом.

- class **Controller**

Атрибуты:

private ResourceBundle resources; // набор ресурсов

private URL location; //Путь до файла, если ввод из файла

private ChoiceBox<Integer> choice_box_number_of_vertexes = new ChoiceBox<>(); //Выбор размера матрицы

private AnchorPane main_pane //контейнер для отобр-я содержимого окна

private Button get_started_button; // начать алгоритм

private Button floydwhat_button; // об алгоритме Флойда-Уоршелла

private Button fileinput_button; //ввод из файла

private Button exit_button; // выйти из приложения

private Button faq_button; // о приложении

private AnchorPane explanation_pane; //контейнер для вкладки об ал горитме

private AnchorPane fileinput_pane; //контейнер для вкладки ввод из файла

private Button choosefile_button; // кнопка выбрать файл

private Button filename_submitbutton; // кнопка подверждения файла

private AnchorPane faq_pane // контейнер для вкладки о приложении

private Button main_sheet_button; //кнопка для главной вкладки

private Label filenamelabel ;// метка для названия файла

Методы:

void exit(ActionEvent event)

Метод осуществляет выход из приложения.

void choose_file_function(ActionEvent event)

Метод отображает диалоговое окно для выбора файла с графом

void filename_submit_button(ActionEvent event)

Метод подтверждающий выбор файла и начинающий его парсинг, а затем проверку на корректность. Если проверка успешна, начинается визуализация алгоритма

private void alert(String alert_message1, String alert_message2)

Метод отображает предупреждение с заголовком alert_message_2 и содержанием alert_message_1

void explanation(ActionEvent event)

Метод отображающий вкладку с объяснением алгоритма

void faq(ActionEvent event)

Метод отображающий вкладку о приложении.

void fileinput(ActionEvent event)

Метод отображающий вкладку с выбором файла для ввода

void main_sheet(ActionEvent event)

Метод отображающий главную вкладку, предлагающую начать ввод матрицы

private void fadeIn(AnchorPane pane)

Анимация при появлении вкладки

private void fadeOut(AnchorPane pane)

Анимация при ее закрытии

public static void callgraph()

Метод запускающий файл buildgraph.jar, предназначенный для отображения графа.

void initialize()

Метод осуществляющий визуальный переход к форме заполнения матрицы после выбора ее размера.

- class **Main**

  Точка вход в приложение

- class **BuildGraph**

public static String read_from_file()

Метод осуществляет считывание матрицы графа из файла.

void addcomponent(int V, int vertex1,int vertex2, int vertex3)

Метод осуществляющий изменение графа в соотвествие с шагом алгоритма.

public void BuildGraphf()

Метод осуществляющий отрисовку графа.

main и служит для отрисовки графа.

3.2. Архитектура приложения

UML – диаграмма реализованных классов представлена на рис. 1.

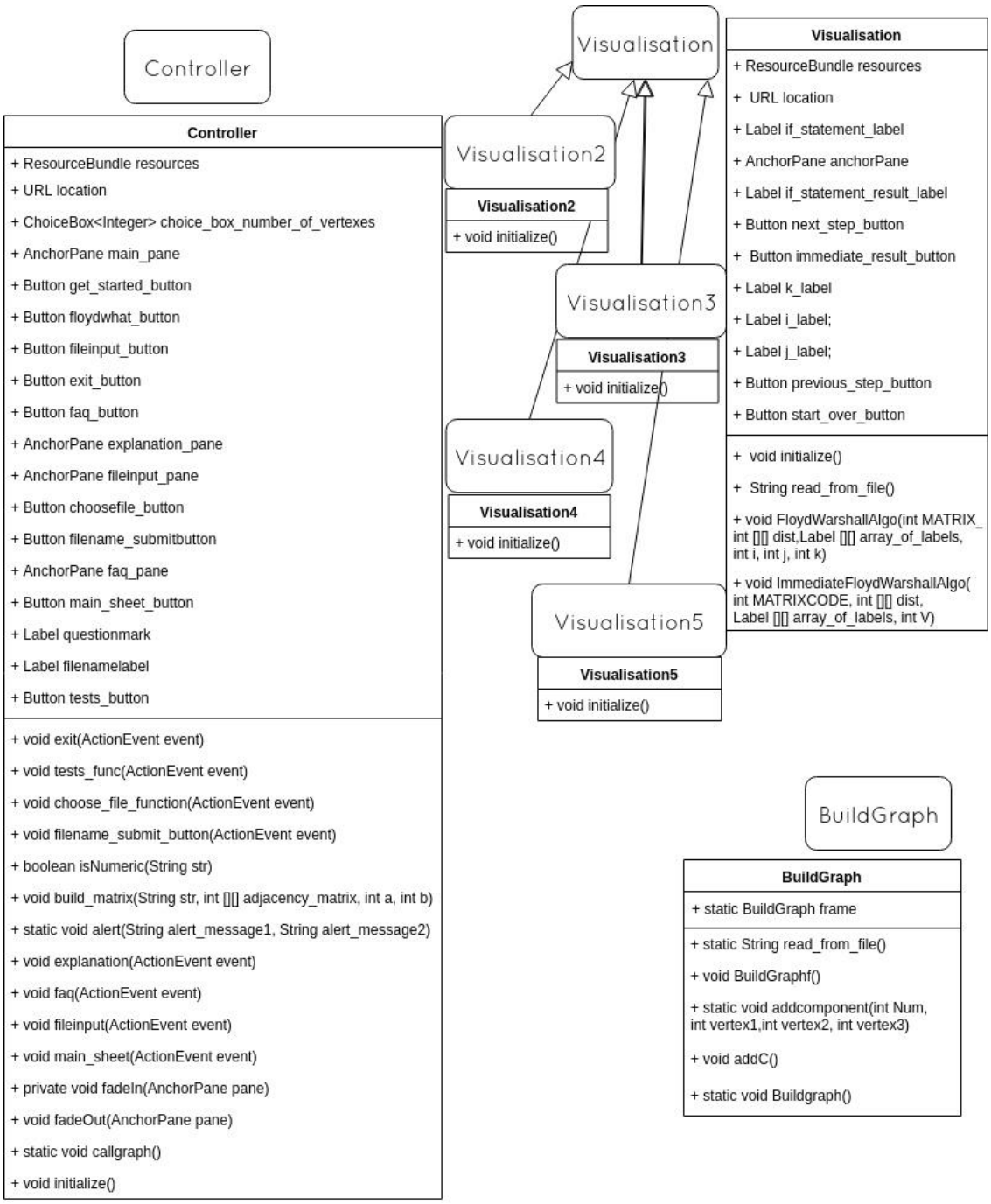


Рисунок 1 – UML – диаграмма

4. ОСОБЕННОСТИ РЕАЛИЗАЦИИ

4.1 Тестирование графического интерфейса

Главное меню содержит 5 вкладок: создание графа, описание алгоритма, выбор файла с графом из txt файла, FAQ и выход. Их вид предоставлен на рис.2-6.

Рисунок 2 – Главное меню

Рисунок 3 – Описание алгоритма Флойда-Уоршелла

Рисунок 4 – Ввод матрицы смежности из файла

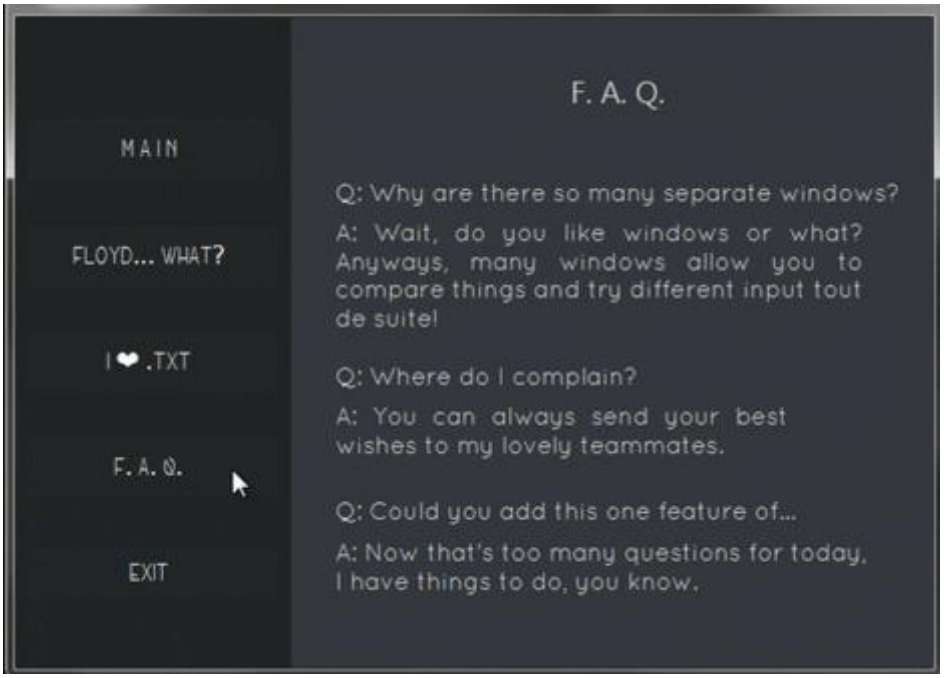


Рисунок 5 – FAQ

Из вкладки с рис.1 осуществляется переход в форму указанную на рис.5, где пользователю предлагается заполнить матрицу смежности графа.

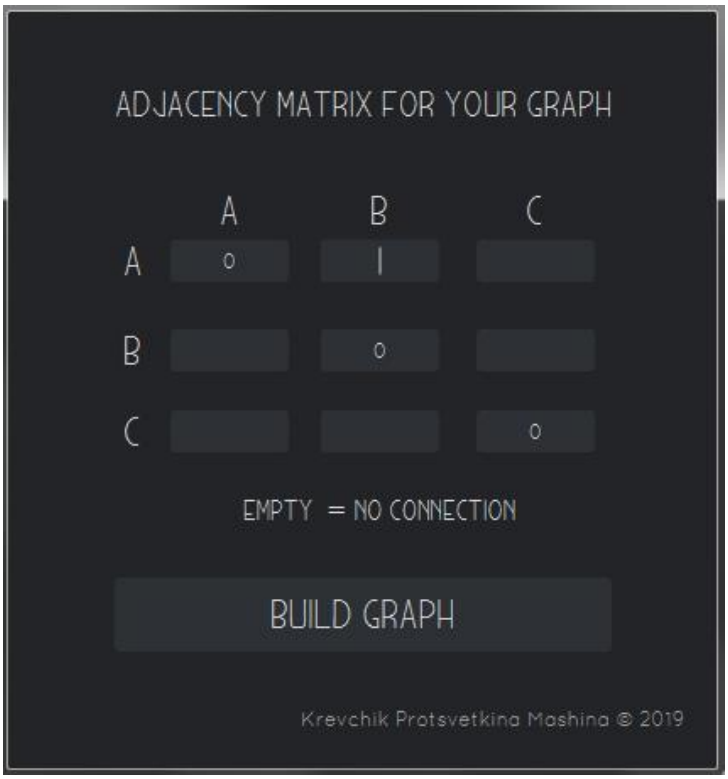


Рисунок 6 – Ввод матрицы смежности

После нажатия кнопки BUILD GRAPH открывается 2 окна, которые приведены на рис.7.

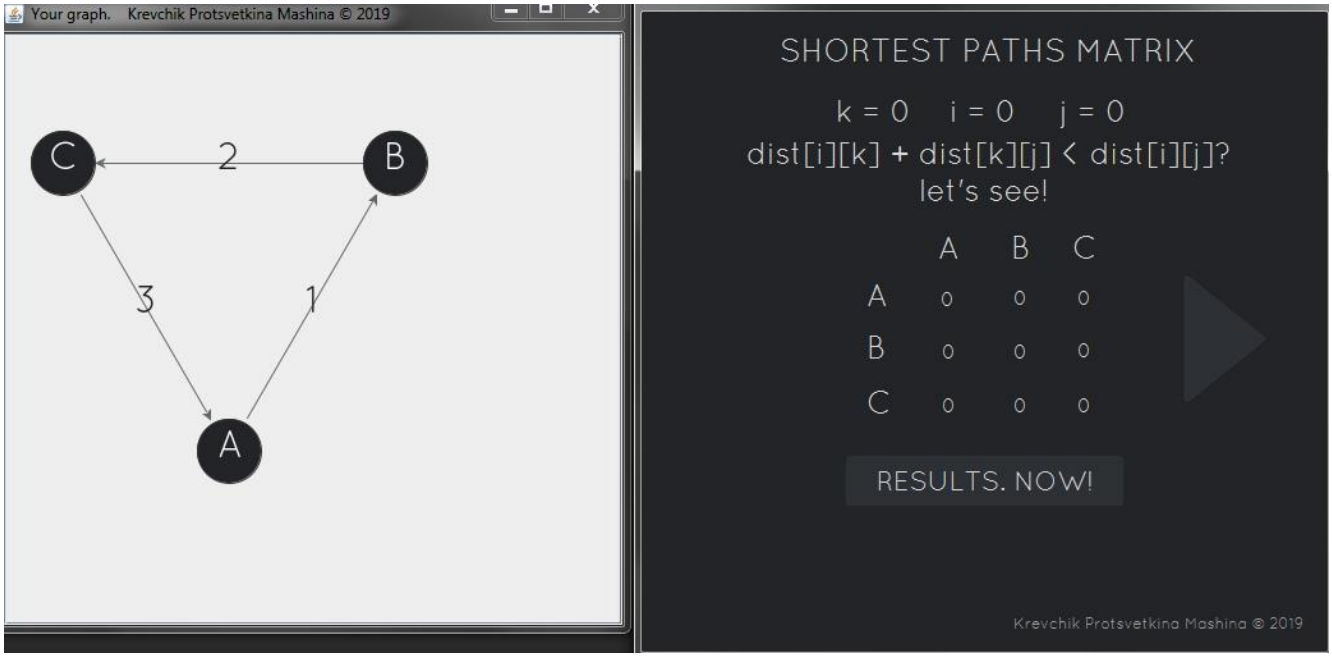


Рисунок 7 – Отображение графа

Мгновенный результат работы алгоритма можно получить, нажав клавишу RESULTS. NOW, а кнопка START OVER запустит его заново.

Очередной шаг алгоритма показан на рис.8

Рисунок 8 – Шаг алгоритма.

4.2 Тестирование ввода из файла

Корректность работы алгоритма была протестирована на 7 файлах с входными данными, 6 из которых были некорректны.

Тест 1: 3 1 1 1 0 0 0 0 0 0 (граф содержит петли)

Результат: рис.9

Тест 2: 3 0 0 1 0 0 1 1 0 0 2 (файл содержит избыточные данные)

Результат: рис. 10

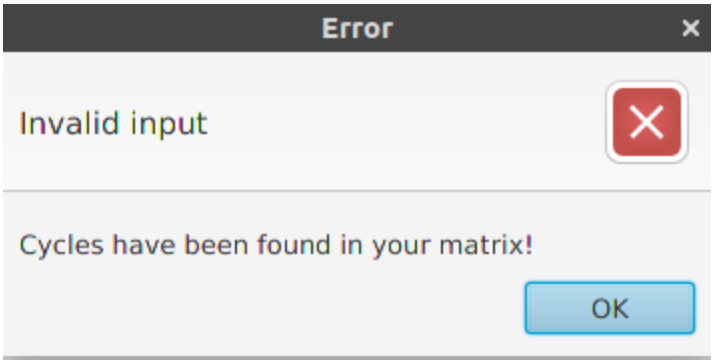


Рисунок 9 — Результат работы программы для графа с циклом

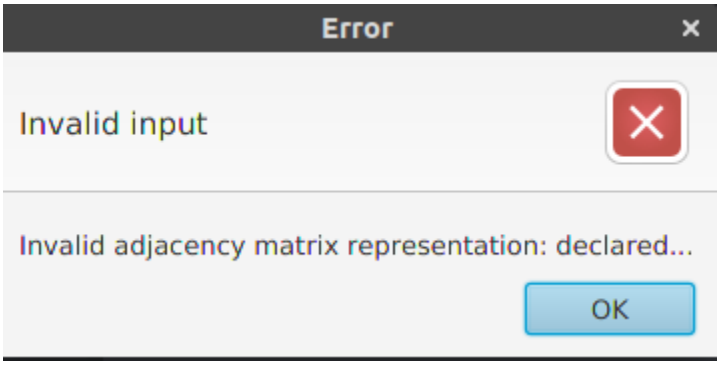
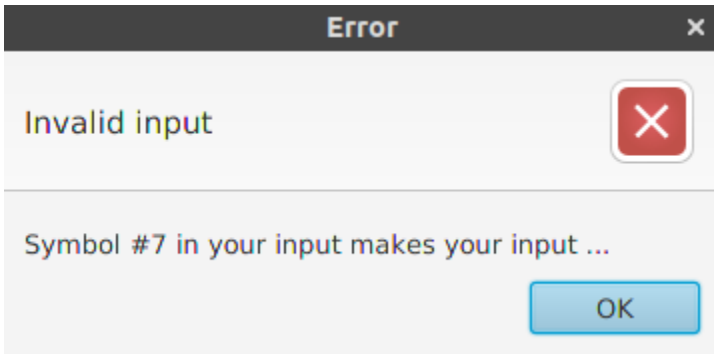


Рисунок 10 — Результат работы программы для файла с избыточными данными

Тест 3: 3 0 1 -1 2 0 0 3 0 0 (граф с отрицательными весами)



Результат: рис. 11

Рисунок 11 – Результат работы программы для графа с отрицательными весами

Тест 4: 3 0 2 4 (недостаточное количество информации)

Результат: рис. 12

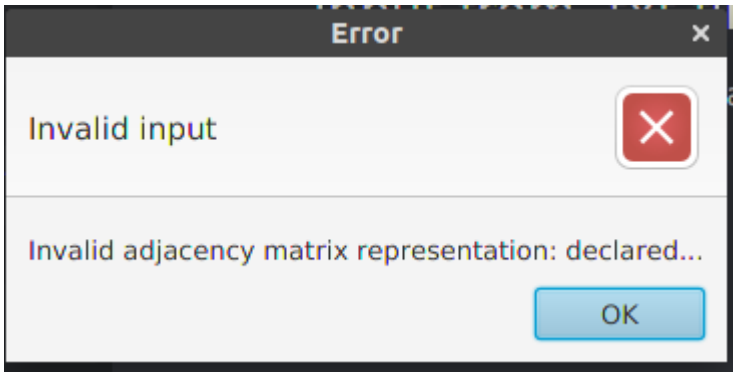


Рисунок 12 – Результат работы программы для файла с недостаточной информацией

Тест 5: 3 0 a 0 3 3 3 3 3 3 (файл содержит символы)

Результат: рис. 13

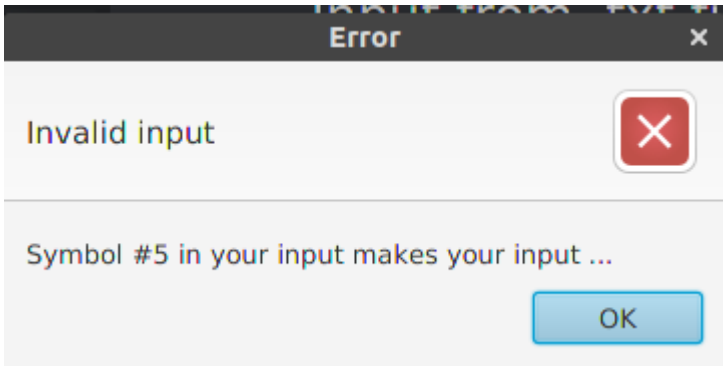


Рисунок 13 – Результат работы программы для файла, содержащего символы

Тест 6: 0 2 2 2 (указан неверный код матрицы)

Результат: рис. 14

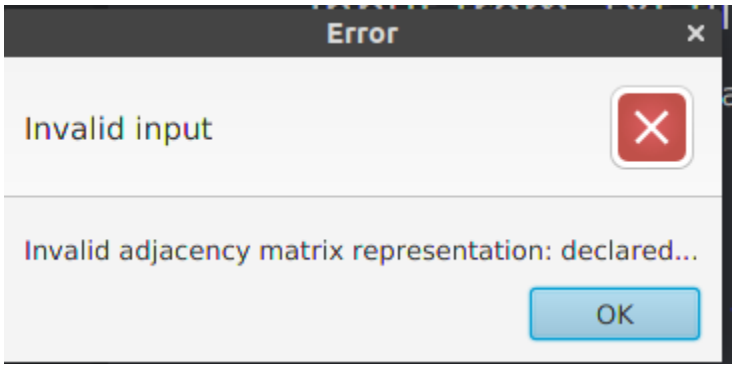


Рисунок 14 – Результат работы программы для файла, содержащего неверный код матрицы.

Тест 7: 3 0 0 9 1 0 3 12 7 0 (корректный)

Результат: рис. 15

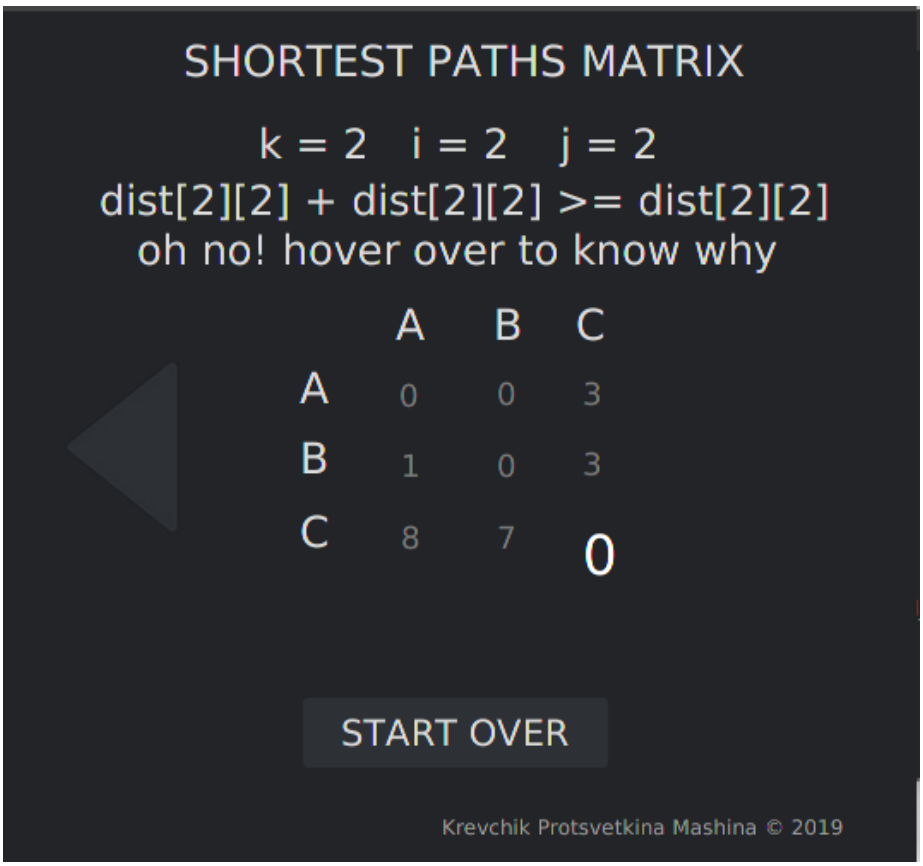


Рисунок 15 – Результат работы алгоритма для верного теста

4.3. Тестирование кода алгоритма

Для тестирования алгоритма Флойда-Уоршелла был создан класс TestClass, в котором создается экземпляр класса с графом, после чего последовательно вызывается метод самого алгоритма Флойда-Уоршелла. Тестовый класс записывает результаты в логовый файл test_log.txt. Содержимое файла с результатом тестов приведено ниже:

[Test 1]

Matrix:

1 2 3

3 3 3

3 3 3

Output:

[Err] Loops found. Return.

[Test 2]

Matrix:

0 -1 0

2 0 3

3 3 0

Output:

[Err] Negative weights found. Return.

[Test 3]

Matrix:

0 0 0

0 0 0

0 0 0

Output:

0 0 0

0 0 0

0 0 0

[Test 4]

Null pointer example:

[Err] Null pointer found. Return.

[Test 5]

Matrix:

0 1000 0

10 0 1010

12 18 0

Output:

0 18 0
10 0 10
12 18 0

ЗАКЛЮЧЕНИЕ

В рамках учебной практической деятельности было разработано GUI-приложение визуализирующее алгоритм Флойда-Уоршалла для поиска кратчайших путей между всеми парами вершин графа. Приложение позволяет запускать алгоритм в пошаговом или мгновенном режиме. Присутствует возможность вернуться на шаг назад или начать алгоритм сначала. Введенный пользователем граф отображается *плоском* (2-d) виде. Сам алгоритм Флойда-Уоршалла отображается в отдельном от графов окне, где на каждом шаге алгоритма выделяются текущие элементы матрицы путей. На каждом шаге алгоритма производится сравнение некоторых числовых величин, которые так же видны пользователю с помощью hover эффекта. Все приложение выполнено в приятной темной цветовой гамме прекрасно сочетающейся с шрифтами Quicksand. Динамичность и интерактивность в совокупности с изящным стилем оформления обеспечивают пользователю комфортную и понятную работу с приложением.

Выполнение данного проекта позволило приобрести навыки, необходимые в сфере IT, а именно:

- программирование на языке Java,
- работа в команде,
- итеративная разработка программного продукта.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1) https://docs.oracle.com/en/java/

2) https://jgraph.github.io/mxgraph/docs/tutorial.html

3) https://www.baeldung.com/jgrapht

ПРИЛОЖЕНИЕ А.
ИСХОДНЫЙ КОД ПРОГРАММЫ

- **файл buildgraph.java**

```
import com.mxgraph.layout.mxcirclelayout;
      import com.mxgraph.layout.mxparalleledgelayout;
      import com.mxgraph.model.mxcell;
      import com.mxgraph.swing.mxgraphcomponent;
      import com.mxgraph.swing.util.mxmorphing;
      import com.mxgraph.util.mxconstants;
      import com.mxgraph.util.mxevent;
      import com.mxgraph.util.mxeventsource;
      import com.mxgraph.view.mxcellstate;
      import com.mxgraph.view.mxgraph;
      import com.mxgraph.view.mxgraphview;

      import java.awt.event.*;
      import javax.swing.*;
      import java.io.*;


      import java.awt.color;
      import java.awt.dimension;
      import java.awt.graphics;
      import java.awt.graphics2d;
      import java.awt.graphicsdevice;
      import java.awt.graphicsenvironment;
      import java.awt.rectangle;
      import java.io.bufferedreader;
      import java.io.file;
      import java.io.filenotfoundexception;
      import java.io.filereader;
      import java.io.ioexception;
      import java.nio.file.filesystems;
      import java.nio.file.path;
      import java.nio.file.standardwatcheventkinds;
      import java.nio.file.watchevent;
      import java.nio.file.watchkey;
      import java.nio.file.watchservice;
      import java.util.arraylist;
      import java.util.hashmap;
      import java.util.map;
      import java.util.scanner;

      import javax.swing.jframe;
```

```
import javax.swing.jpanel;
public class buildgraph extends jframe
{

    /**
     *
     */
    //jframe.setbackground( color.getcolor("#cdcdcd") );
    // private static final long serialversionuid = -
2707712944901661771l;
    final static int inf = 100000, frame_width = 500,frame_height =
500;


    public  static string read_from_file()  { //read the file with
adjacency matrix input from gui
        file file = new file("adjacency_matrix.txt");


        bufferedreader br = null;
        try {
            br = new bufferedreader(new filereader(file));
        } catch (filenotfoundexception e) {
            e.printstacktrace();
        }

        string st = null; //system.out.println(st);
        try {
            st = br.readline();
        } catch (ioexception e) {
            e.printstacktrace();
        }
        return st;
        //system.out.println(st);
        //system.out.println("hello world!");
    }

    mxgraph graph = new mxgraph();
    mxgraphcomponent graphcomponent = new mxgraphcomponent(graph);
    object [] array_of_vertexes ;
    object [][] array_of_edges ;
    static int v = 0;

static int flag = 0;
```

```
          public void buildgraphf(){

             // super("your graph.    krevchik protsvetkina mashina © 2019");


              object parent = graph.getdefaultparent();

              string matrix = read_from_file();
              var arrayofstrings = matrix.split(" ");//turn input line from file into an array of basically "integers",
              //but they are in a form of strings. we'll parse them into an array of integers right away

              //mxmorphing();
              int v = integer.parseint(arrayofstrings[0]);
              int input_array[] = new int [arrayofstrings.length-1];

              for (int i = 0; i< arrayofstrings.length-1; i++){
                  input_array[i]= integer.parseint(arrayofstrings[i+1]);
              }


              array_of_vertexes = new  object[v];
              array_of_edges = new  object[v][v];

              int adj_matrix[][] = new int [v][v];

              int input_array_iterator = 0;
              for(int q = 0; q<v; q++)
                  for(int j= 0;j<v;j++){
                      adj_matrix[q][j] = input_array[input_array_iterator];
                      input_array_iterator++;
                  }




              //////////
              graph.getmodel().beginupdate();
              //////////////
```

```
            int x = 0, y = 0;


            try
            {

                for(int w = 0; w< v; w++){

                    if (w%4==0){
                        x = 50; y += 100;
                    }
                    else if(w%4==1){
                        x=300;
                    }
                    else if(w%4==2){
                        x=100; y+=100;
                    }
                    else if(w%4==3){
                        x=400;
                    }

                    array_of_vertexes[w] = graph.insertvertex(parent,
null, (char)('a'+w), x, y, 50,50,
"shape=ellipse;strokecolor=#000000;fillcolor=#123456");
                    graph.setcellstyles(mxconstants.style_fillcolor,
"#222427", new object[]{array_of_vertexes[w]});
                    graph.setcellstyles(mxconstants.style_fontcolor,
"#ffffff", new object[]{array_of_vertexes[w]});
                    graph.setcellstyles(mxconstants.default_fontfamily,
"quicksand-regular", new object[]{array_of_vertexes[w]});



                }


                mxconstants.default_fontfamilies = "quicksand-regular";
//default_fontfamilies("calibri light");
                mxconstants.default_fontfamily = "quicksand-regular";
                mxconstants.default_fontsize = 30;


                int counter = 0;
```

```
                for(int q = 0; q<v; q++){
                    for(int j= 0;j<v;j++){
                        if(adj_matrix[q][j]<inf&&q!=j){
                            array_of_edges[q][j] =
graph.insertedge(parent, null, adj_matrix[q][j], array_of_vertexes[q],
array_of_vertexes[j]);

graph.setcellstyles(mxconstants.style_strokecolor, "#636668", new
object[]{array_of_edges[q][j]});

graph.setcellstyles(mxconstants.style_fontcolor, "#222427", new
object[]{array_of_edges[q][j]});

graph.setcellstyles(mxconstants.style_strokewidth, "2", new
object[]{array_of_edges[q][j]});
                            //graph.gettooltipforcell

                            counter++;
                        }
                    }
                }

                system.out.printf("'%d'", counter);
            }
            finally
            {
                graph.getmodel().endupdate();
            }


            graph.setcellseditable(false);
            graph.setallowdanglingedges(false);
            graph.setallowloops(false);
            graph.setcellsdeletable(false);
            graph.setcellscloneable(false);
            graph.setcellsdisconnectable(false);
            graph.setdropenabled(false);
            graph.setsplitenabled(false);
            graph.setcellsbendable(false);
            graph.setdisconnectonmove(false);
            graph.setdropenabled(false);
            graph.setallownegativecoordinates(false);
            graph.setswimlanenesting(true);
            graph.setcellscloneable(false);
```

```
            graph.setcellsdisconnectable(false);
            graph.setallowloops(false);
            graph.setsplitenabled(false);
            graph.setcellseditable(false);
            graph.setallowdanglingedges(false);


            graph.setenabled(false);

            int o = 0;


            new mxcirclelayout(graph,
200).execute(graph.getdefaultparent());
            new mxparalleledgelayout(graph,
30).execute(graph.getdefaultparent()); // redo for different matrices
!!!!!!


            mxgraphcomponent graphcomponent2 = new
mxgraphcomponent(graph);
            graphcomponent = graphcomponent2;


        }

        void addcomponent(int v, int vertex1,int vertex2, int vertex3){
            for(int q = 0; q<v; q++){
                for(int j= 0;j<v;j++){
                    graph.setcellstyles(mxconstants.style_strokecolor,
"#636668", new object[]{array_of_edges[q][j]});
                    graph.setcellstyles(mxconstants.style_fontcolor,
"#222427", new object[]{array_of_edges[q][j]});
                }
                graph.setcellstyles(mxconstants.style_fillcolor,
"#222427", new object[]{array_of_vertexes[q]});
                graph.setcellstyles(mxconstants.style_fontcolor,
"#ffffff", new object[]{array_of_vertexes[q]});
            }

            graph.setcellstyles(mxconstants.style_fontcolor, "#000000",
new object[]{array_of_vertexes[vertex1]});
            graph.setcellstyles(mxconstants.style_fontcolor, "#000000",
new object[]{array_of_vertexes[vertex2]});
```

```
            graph.setcellstyles(mxconstants.style_fontcolor, "#000000",
new object[]{array_of_vertexes[vertex3]});

            graph.setcellstyles(mxconstants.style_fillcolor, "#78b903",
new object[]{array_of_vertexes[vertex1]});
            graph.setcellstyles(mxconstants.style_fillcolor, "#78b903",
new object[]{array_of_vertexes[vertex2]});
            graph.setcellstyles(mxconstants.style_fillcolor, "#78b903",
new object[]{array_of_vertexes[vertex3]});
            //graph.setcellstyles(mxconstants.style_fontcolor,
"#ffffff", new object[]{array_of_vertexes[w]});
            graph.setcellstyles(mxconstants.style_strokecolor,
"#457202", new object[]{array_of_edges[vertex1][vertex2]});
            graph.setcellstyles(mxconstants.style_fontcolor, "#263f00",
new object[]{array_of_edges[vertex1][vertex2]});
            graph.setcellstyles(mxconstants.style_strokecolor,
"#457202", new object[]{array_of_edges[vertex2][vertex3]});
            graph.setcellstyles(mxconstants.style_fontcolor, "#263f00",
new object[]{array_of_edges[vertex2][vertex3]});
            graph.setcellstyles(mxconstants.style_strokecolor,
"#457202", new object[]{array_of_edges[vertex1][vertex3]});
            graph.setcellstyles(mxconstants.style_fontcolor, "#263f00",
new object[]{array_of_edges[vertex1][vertex3]});
            //array_of_edges[vertex2][vertex3].
           // mxgraphview view = new mxgraphview(graph);
          // map<string,object> map = new hashmap<string, object>();
           // map.put("fontcolor=#263f00",new
object[]{array_of_edges[vertex2][vertex3]});
                    // mxcellstate state = new mxcellstate(view,
(mxcell)array_of_edges[0][1], map);
            //state.


            graphcomponent.refresh(); // perhaps unnecessary

            graph.refresh();// perhaps unnecessary
        }


        void addc(){
            getcontentpane().add(graphcomponent);
        }
```

```
          public static void main(string[] args) throws ioexception,
interruptedexception {

              buildgraph frame = new buildgraph();


              frame.setdefaultcloseoperation(jframe.exit_on_close);
              frame.setsize(frame_width, frame_height);
              frame.setlocationrelativeto(null);

      //frame.add(graph);
              frame.buildgraphf();

              frame.addc();
              //frame.addcomponent(0,1);


              frame.setvisible(true);

              //frame.setvisible(true);
      int aa=0;
              final path path =
filesystems.getdefault().getpath(system.getproperty("user.dir"));
              //system.out.println(path);
              try (final watchservice watchservice =
filesystems.getdefault().newwatchservice()) {
                  final watchkey watchkey = path.register(watchservice,
standardwatcheventkinds.entry_modify);
                  while (true) {
                      final watchkey wk = watchservice.take();
                      for (watchevent<?> event : wk.pollevents()) {
                          //we only register "entry_modify" so the
context is always a path.
                          final path changed = (path) event.context();
                          //system.out.println(changed);
                          if (changed.endswith("changes.txt")) {
                              //system.out.println("change");
                              string st =""; file file = new
file("changes.txt");
                              try{st = new scanner(file)
                                      .usedelimiter("\\a").next();}
                              catch (exception e){

                              }
```

```
                                string [] starr = st.split(" ");
                                system.out.printf("'%s'\n", st);
                                for (int r = 0; r< starr.length;r++){
                                    system.out.printf("'%s'\n", starr[r]);
                                }

                               // char [] array = st.tochararray();

                                frame.addcomponent(
integer.valueof(starr[0]),integer.valueof(starr[1]),integer.valueof(starr
[2]),integer.valueof(starr[3])); aa++;


                            }
                        }
                        // reset the key
                        boolean valid = wk.reset();
                        if (!valid) {
                            system.out.println("key has been
unregistered");
                        }
                    }
                }

            }

        }
```

- файл main.java

```
package floydwarshall;

      import javafx.application.application;
      import javafx.fxml.fxmlloader;
      import javafx.scene.parent;
      import javafx.scene.scene;
      import javafx.stage.stage;

      import java.io.file;


      public class main extends application {

          @override
          public void start(stage primarystage) throws exception{
              parent root =
fxmlloader.load(getclass().getresource("frames/menu.fxml"));
              primarystage.settitle("main menu");
              primarystage.setscene(new scene(root, 600, 425));
              primarystage.setresizable(false);
              primarystage.show();
          }


          public static void main(string[] args) {
              file file = new file("adjacency_matrix.txt");
              file.delete();

              launch(args);


          }
      }
```

- файл visualisation.java

```
package floydwarshall;

      import java.io.*;
      import javafx.scene.layout.anchorpane;
      import javafx.scene.text.font;
      import javafx.scene.paint.color;
      import java.net.url;
      import java.util.resourcebundle;
      import javafx.fxml.fxml;
      import javafx.scene.control.button;
      import javafx.scene.control.label;
      import javafx.scene.control.tooltip;

      public class visualisation {

          @fxml
          private resourcebundle resources;

          @fxml
          private url location;

          @fxml
          private label if_statement_label;

          @fxml
          private anchorpane anchorpane;

          @fxml
          private label if_statement_result_label;

          @fxml
          private label aa_connection_label = new label();

          @fxml
          private label ab_connection_label = new label();

          @fxml
          private label ac_connection_label = new label();

          @fxml
          private label ba_connection_label = new label();
```

```
@fxml
private label bb_connection_label = new label();

@fxml
private label bc_connection_label = new label();

@fxml
private label ca_connection_label = new label();

@fxml
private label cb_connection_label = new label();

@fxml
private label cc_connection_label = new label();

@fxml
private button next_step_button;

@fxml
private button immediate_result_button;

@fxml
private label k_label;

@fxml
private label i_label;

@fxml
private label j_label;

@fxml
private button previous_step_button;

@fxml
private button start_over_button;



@fxml
void initialize() {



    final int matrix_code = 3;
```

```
            label [][] array_of_labels = {
                    {aa_connection_label,ab_connection_label,
ac_connection_label},

{ba_connection_label,bb_connection_label,bc_connection_label},

{ca_connection_label,cb_connection_label,cc_connection_label}
            };



            previous_step_button.setvisible(false);
            start_over_button.setvisible(false);


            tooltip tp = new tooltip(string.format("always checking the
following:\ndist[i][k] + dist[k][j] < dist[i][j]"));
            tooltip.install(if_statement_label, tp);

            string matrix = read_from_file();
            string [] arrayofstrings = matrix.split(" ");//turn input
line from file into an array of basically "integers",
            //but they are in a form of strings. we'll parse them into
an array of integers right away
            int v = integer.parseint(arrayofstrings[0]);
            int input_array[] = new int [arrayofstrings.length-1];

            for (int i = 0; i< arrayofstrings.length-1; i++){
                input_array[i]= integer.parseint(arrayofstrings[i+1]);
            }

            int adj_matrix[][] = new int [v][v];


            int input_array_iterator = 0;
            for(int q = 0; q<v; q++)
                for(int j= 0;j<v;j++){
                    adj_matrix[q][j] =
input_array[input_array_iterator];
                    input_array_iterator++;
                }
```

```
            for(int z = 0; z < v; z++)
                for(int x = 0; x< v; x++){
                    array_of_labels[z][x].settooltip(new
tooltip(string.format("dist [%d][%d]", z,x)));
                }


            int dist[][] = new int[v][v]; //new adjacency matrix
            //int i, j, k;

            for (int i = 0; i < v; i++)
                for (int j = 0; j < v; j++)
                    dist[i][j] = adj_matrix[i][j];


            immediate_result_button.setonaction(event -> {
                //changes();
                previous_step_button.setvisible(true);
                start_over_button.setvisible(true);
                immediatefloydwarshallalgo(matrix_code,dist,
array_of_labels,v);
                immediate_result_button.setvisible(false);
                next_step_button.setvisible(false);
            });

            next_step_button.setonaction(event -> {

                //changes();

                immediate_result_button.setvisible(true);
                previous_step_button.setvisible(true);
                start_over_button.setvisible(true);

                boolean increase_j = true;
                if (k<v) {
                    //system.out.printf("%d %d %d\n",k,i,j);
                    floydwarshallalgo(matrix_code, dist,
array_of_labels, i, j, k);

                    if (i == v - 1&& j == v-1) {
                        k++;
                        j = 0;
                        i = 0;
                        increase_j = false;
```

```
                }
                if (j == v - 1) {//
                    i++;
                    j = 0;//
                    increase_j = false;
                }
                if (increase_j) j++;

            }

            else{
            immediate_result_button.setvisible(false);
            next_step_button.setvisible(false);
            }

        });

        start_over_button.setonaction(event -> {
           // changes();
            previous_step_button.setvisible(false);
            start_over_button.setvisible(false);
            for (int q = 0; q < v; q++)
                for (int w = 0; w < v; w++)
                    dist[q][w] = adj_matrix[q][w];
            i=0; k =0; j = 0;
            immediate_result_button.setvisible(true);
            next_step_button.setvisible(true);
            floydwarshallalgo(matrix_code, dist, array_of_labels,
i, j, k);
        });

        previous_step_button.setonaction(event -> {
           // changes();
            immediate_result_button.setvisible(true);
            start_over_button.setvisible(true);
            next_step_button.setvisible(true);
            for (int q = 0; q < v; q++)
                for (int w = 0; w < v; w++)
                    dist[q][w] = adj_matrix[q][w];

            if(i==v && j == v && k == v){
                i--; j--; k--;
            }
            else {
```

```
                    if (j != 0) {
                        j--;

                    } else if (i != 0) {
                        i--;
                        j = v - 1;

                    } else if (k != 0) {
                        i = v - 1;
                        j = v - 1;
                        k--;

                    }
                }
                int k2 = k, i2 = i, j2 = j; //backup for ijk
                k = 0; j = 0; i = 0;

                while(i!=i2||j!=j2||k!=k2){

                    boolean increase_j = true;
                    if (k<v) {

                        floydwarshallalgo(matrix_code, dist,
array_of_labels,i, j, k);

                        if (i == v - 1&& j == v-1) {
                            k++;
                            j = 0;
                            i = 0;
                            increase_j = false;
                        }
                        if (j == v - 1) {//
                            i++;
                            j = 0;//
                            increase_j = false;
                        }
                        if (increase_j) j++;

                    }

                }

                if(i==0 && j==0 && k==0){
```

```
                    previous_step_button.setvisible(false);
                    start_over_button.setvisible(false);
                }

            });

        }

        /* public void changes(){
            try (filewriter writer = new filewriter("changes.txt",
false)) {

                writer.append("");

                writer.flush();
            } catch (ioexception ex) {

                system.out.println(ex.getmessage());
            }

        }*/

        static int i=0,j=0,k=0; //////

        final static int inf = 100000;
        protected  string read_from_file()  { //read the file with
adjacency matrix input from gui
            file file = new file("adjacency_matrix.txt");


            bufferedreader br = null;
            try {
                br = new bufferedreader(new filereader(file));
            } catch (filenotfoundexception e) {
                e.printstacktrace();
            }

            string st = null; //system.out.println(st);
            try {
                st = br.readline();
            } catch (ioexception e) {
                e.printstacktrace();
            }
            return st;
```

```
        }


        protected void floydwarshallalgo(int matrix_code, int [][]
dist,label [][] array_of_labels, int i, int j, int k) {

            for(int q = 0; q<matrix_code;q++)
                for(int w = 0;w<matrix_code;w++) {

array_of_labels[q][w].settextfill(color.web("#7b7b7b"));
                    array_of_labels[q][w].setfont(new font("calibri
light", 18));
                }

            {try (filewriter writer = new filewriter("changes.txt",
false)) {

                writer.append(integer.tostring(matrix_code));
                writer.append(' ');
                writer.append(integer.tostring(i));
                writer.append(' ');
                writer.append(integer.tostring(k));
                writer.append(' ');
                writer.append(integer.tostring(j));

                writer.flush();
            } catch (ioexception ex) {

                system.out.println(ex.getmessage());
            }}


            array_of_labels[i][k].setfont(new font("calibri", 32));
            array_of_labels[k][j].setfont(new font("calibri", 32));
            array_of_labels[i][j].setfont(new font("calibri", 32));


            array_of_labels[i][k].settextfill(color.web("#ffffff"));
            array_of_labels[k][j].settextfill(color.web("#ffffff"));
            array_of_labels[i][j].settextfill(color.web("#ffffff"));

            k_label.settext(string.format("k = %d", k));
            i_label.settext(string.format("i = %d", i));
```

```
            j_label.settext(string.format("j = %d", j));


            if (dist[i][k] + dist[k][j] < dist[i][j]) {
                if_statement_label.settext(string.format("dist[%d][%d]
+ dist[%d][%d] < dist[%d][%d]", i, k, k, j, i, j));
                dist[i][j] = dist[i][k] + dist[k][j];
                if_statement_result_label.settext(string.format("
dist[%d][%d] = %d = %d + %d", i, j, dist[i][j], dist[i][k], dist[k][j]));
                tooltip tp = new tooltip(string.format("dist[%d][%d] =
dist[%d][%d] + dist[%d][%d]", i, j, i, k, k, j));
                tooltip.install(if_statement_result_label, tp);

            } else {
                if_statement_result_label.settext("oh no! hover over to
know why");
                stringbuilder str = new stringbuilder();
                if (dist[i][k] + dist[k][j]>=inf)
                    str.append(string.format("dist[%d][%d] +
dist[%d][%d] = ∞\nand it's not less than ", i, k, k, j));
                else
                    str.append(string.format("dist[%d][%d] +
dist[%d][%d] = %d\nand it's not less than ", i, k, k, j, dist[i][k] +
dist[k][j]));
                if (dist[i][j]>=inf)
                    str.append(string.format("dist[%d][%d] = ∞!", i,
j));
                else
                    str.append(string.format("dist[%d][%d] = %d!", i,
j, dist[i][j]));

                tooltip tp = new tooltip(str.tostring());
                tooltip.install(if_statement_result_label, tp);
                if_statement_label.settext(string.format("dist[%d][%d]
+ dist[%d][%d] >= dist[%d][%d]", i, k, k, j, i, j));
            }

            for(int z = 0; z < matrix_code; z++){

                for(int x = 0; x < matrix_code; x++){
                    if (z==x)
                        continue;
                    if (dist[z][x] < inf)
array_of_labels[z][x].settext(string.valueof(dist[z][x]));
```

```
                    else {array_of_labels[z][x].settext("∞");

                    }
                }
            }

        }

        protected void immediatefloydwarshallalgo(int matrixcode, int
[][] dist,label [][] array_of_labels, int v){
            for (k = 0; k < v; k++)
            {
                for (i = 0; i < v; i++)
                {
                    for (j = 0; j < v; j++)
                    {
                        floydwarshallalgo(matrixcode, dist,
array_of_labels, i,j,k);
                    }
                }
            }
        }



    }
```