Министерство образования и науки РФ

Федеральное государственное бюджетное образовательное учреждение высшего профессионального образования “Санкт-Петербургский государственный электротехнический университет “ЛЭТИ” им. В.И. Ульянова (Ленина)” (СПбГЭТУ)

Кафедра Математического Обеспечения и Применения ЭВМ (МО ЭВМ)

Отчет по проекту

на тему: **«Кратчайшие пути в графах. Алгоритм Дейкстры»**

Выполнил: Сабынин Е.С.

Факультет: КТИ

Группа: 3304

Проверил: Фирсов М. А.

г. Санкт-Петербург

2015 Год

**Исходное задание**

Кратчайшие пути в графе (Dijkstra):визуализация

**Формальная постановка задачи:**

Задан исходный ориентированный граф G=(V, E), где V – количество вершин в графе, для данного графа, в котором каждая вершина пронумерована от 0 до V. Необходимо реализовать алгоритм Дейскстры для данного графа, попутно визуализируя алгоритм.

**Описание алгоритма:**

Алгоритм Дейкстры решает задачу о кратчайших путях из одной вершины для взвешенного ориентированного графа *G = (V, E)*с исходной вершиной *s*, в котором веса всех рёбер неотрицательны (*http://comp-science.narod.ru/KPG/Path.files/w.jpg(u, v) ≥ 0*для всех *(u, v)http://comp-science.narod.ru/KPG/Path.files/e.jpgE*).

**Формальное объяснение**

В процессе работы алгоритма Дейкстры поддерживается множество S http://comp-science.narod.ru/KPG/Path.files/3.jpg V, состоящее из вершин *v*, для которых *δ(s, v)* уже найдено. Алгоритм выбирает вершину *u http://comp-science.narod.ru/KPG/Path.files/e.jpg V\S* с наименьшим *d*[*u*], добавляет *u* к множеству *S* и производит релаксацию всех рёбер, выходящих из *u*, после чего цикл повторяется.

**Не формальное объяснение**

Каждой вершине из *V* сопоставим метку — минимальное известное расстояние от этой вершины до *a*. Алгоритм работает пошагово — на каждом шаге он «посещает» одну вершину и пытается уменьшать метки. Работа алгоритма завершается, когда все вершины посещены.

**Инициализация.** Метка самой вершины *a* полагается равной *0*, метки остальных вершин — бесконечности. Это отражает то, что расстояния от *a* до других вершин пока неизвестны. Все вершины графа помечаются как непосещенные.

**Шаг алгоритма.** Если все вершины посещены, алгоритм завершается. В противном случае из еще не посещенных вершин выбирается вершина *u*, имеющая минимальную метку. Мы рассматриваем всевозможные маршруты, в которых *u* является предпоследним пунктом. Вершины, соединенные с вершиной *u* ребрами, назовем соседями этой вершины. Для каждого соседа рассмотрим новую длину пути, равную сумме текущей метки *u* и длины ребра, соединяющего *u* с этим соседом. Если полученная длина меньше метки соседа, заменим метку этой длиной. Рассмотрев всех соседей, пометим вершину *u* как посещенную и повторим шаг.

**Спецификация программы:**

В работе будет реализована визуализация алгоритма Дейкстры, с ее помощью пользователь сможет детально увидеть работу поиска кратчайших путей до всех точек, проследить за ходом выбора текущей точки и разобраться в логике алоритма. Визуализация подразумевает наличие графического интерфейса, в котором содержится текущее состояние(изображение) направленного взвешенного графа.

**UML-диаграмма:**





**Структура проекта:**

Класс GraphManager предназначен для создания графа, путем чтения из файла информации, также в нем будут храниться данные о графе.

Класс DeikstraWorker предназначен для расчета минимальных путей до каждой вершины. Данный класс будет реализован с использованием паттерна проектирования Singleton(lazyinitialization).С помощью класса GraphManager текущий класс получает доступ к информации о заданном графе. А с помощью функции startDeikstra рассчитает минимальные стоимости до остальных вершин.

Класс Pair реализует структуру данных пара.

Класс FormWorker содержит описание компонент формы.

**Информация о графе в классе содержит:**

ArrayList<ArrayList<Pair<Integer, Integer>>> table; таблица стоимостей ребер из каждой вершины

ArrayList<Character> isVisit; список посещенности вершин

ArrayList<Integer> distance; список стоимостей до вершин

**Входные данные:**

Визуализация подразумевает под собой отрисовку небольших по размерам графов с целью показать пошаговое выполнение алгоритма. Входные данные содержатся в исходном файле “input.txt”, в котором информация о графах представлена в виде

Формат чтения информации из файла:

Input.txt

4 //количество вершин

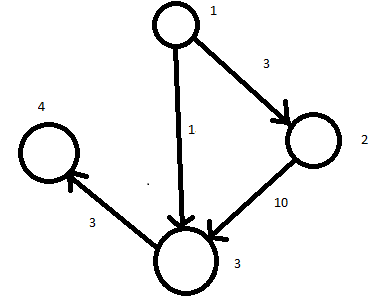
1 2 3 // из вершины 1 в вершину 2, направленное ребро со стоимостью 3

1 3 1

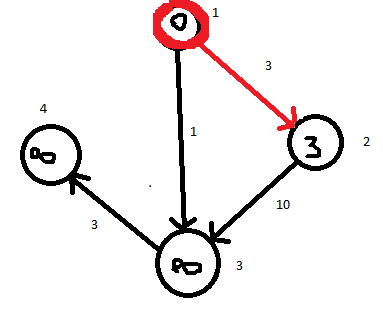
2 3 10

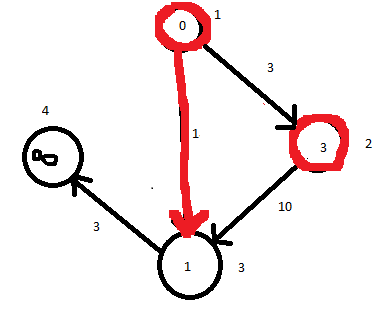
3 4 3

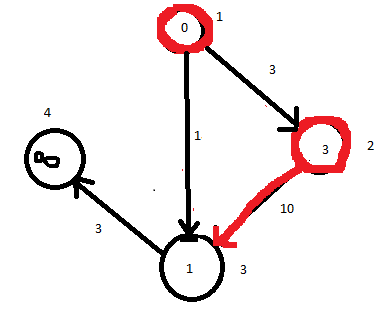
В результате чтения данных из файла получится такой граф.

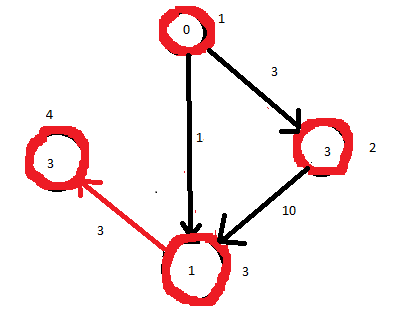


При нажатии клавиши Следующий шаг осуществляется переход к следующему состоянию графа, обновляются дополнительные пометки вершины. Вершины, для которых рассмотрены все входящие ребра окрашиваются в красный цвет. Номера вершин – синие, значения меток – красные. После завершения последнего шага, пометки вершин будут обозначать минимальную стоимость из начальной вершины до данной вершины.





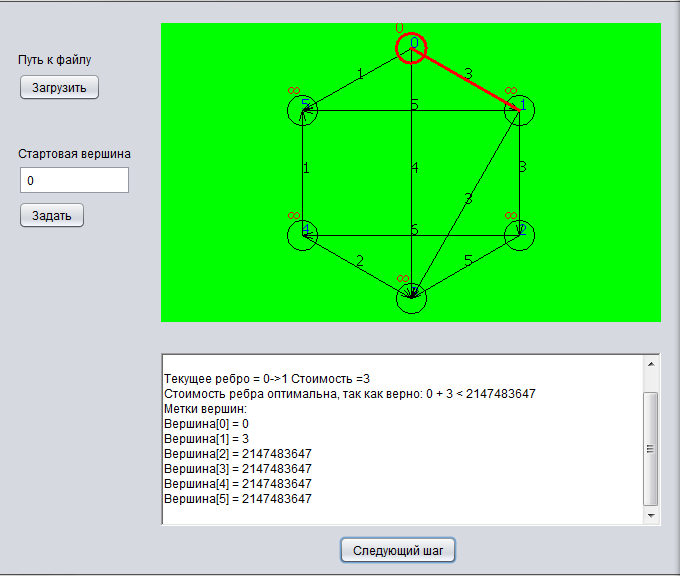




*Выходные данные:*

Пользователь во время выполнения программы может увидеть следующую информацию:

1. форма визуализации



Затем будет выдано сообщение, что работа алгоритма закончена.

Пример:

Алгоритм завершил работу!

Стоимость из вершины 1 в 2 = 3

Стоимость из вершины 1 в 3 = 1

Стоимость из вершины 1 в 4 = 4

**План разработки**

|  |  |
| --- | --- |
| Дата | Что должно быть готово |
| 1 неделя | |
| Среда | Внешняя и внутренняя спецификация, план разработки |
| Четверг | План разработки выложить в репозиторий |
| Пятница | Прототип графического приложения |
| 2 неделя | |
| Понедельник | 1 версия приложения  Необходимо:  Отрисовать граф после чтения из файла  Переход к следующему шагу алгоритма |
| Среда | 2 версия приложения  Обработать исключительные ситуации  Выбор стартовой вершины  Внешние доработки графического интерфейса (кнопки, панели)  Мануал по работе с программой  Промежуточные данные и пометки вершин  Добавить в репозиторий отчет и проект  Должно быть сдано/показано. |

**Код программы:**

**Main.java**

/\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

/\*\*

\*

\* @author Zhenya

\*/

public class Main {

public static void main(String[] args) {

/\* Set the Nimbus look and feel \*/

//<editor-fold defaultstate="collapsed" desc=" Look and feel setting code (optional) ">

/\* If Nimbus (introduced in Java SE 6) is not available, stay with the default look and feel.

\* For details see http://download.oracle.com/javase/tutorial/uiswing/lookandfeel/plaf.html

\*/

try {

for (javax.swing.UIManager.LookAndFeelInfo info : javax.swing.UIManager.getInstalledLookAndFeels()) {

if ("Nimbus".equals(info.getName())) {

javax.swing.UIManager.setLookAndFeel(info.getClassName());

break;

}

}

} catch (ClassNotFoundException ex) {

java.util.logging.Logger.getLogger(FormWorker.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);

} catch (InstantiationException ex) {

java.util.logging.Logger.getLogger(FormWorker.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);

} catch (IllegalAccessException ex) {

java.util.logging.Logger.getLogger(FormWorker.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);

} catch (javax.swing.UnsupportedLookAndFeelException ex) {

java.util.logging.Logger.getLogger(FormWorker.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);

}

//</editor-fold>

/\* Create and display the form \*/

java.awt.EventQueue.invokeLater(new Runnable() {

public void run() {

FormWorker form = new FormWorker();

form.setVisible(true);

}

});

}

}

**Pair.java**

/\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

/\*\*

\*

\* @author Zhenya

\*/

class Pair<T0, T1> {

private T0 first;

private T1 second;

public Pair(T0 first, T1 second){

this.first = first;

this.second = second;

}

public T0 getFirst(){

return first;

}

public T1 getSecond(){

return second;

}

}

**GraphManager.java**

import java.io.File;

import java.io.FileNotFoundException;

import java.util.\*;

/\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

/\*\*

\*

\* @author Zhenya

\*/

public class GraphManager {

public int startPoint;

public int endPoint;

private int edgeCount;

public int pointCount;

public LinkedList<Boolean> isVisit;

public ArrayList<Integer> distance;

public ArrayList<Integer> prevPoints;

public ArrayList<ArrayList<Pair<Integer,Integer>>> table;

public ArrayList<Integer> edgesNumbers;

public GraphManager(int startPoint) {

this.startPoint = startPoint;

}

public void readFromFile(File file) throws FileNotFoundException{

Scanner scanner = new Scanner(file);

edgeCount = Integer.parseInt(scanner.next());

pointCount = Integer.parseInt(scanner.next());

endPoint = pointCount - 1;

table = new ArrayList<ArrayList<Pair<Integer,Integer>>>(pointCount);

isVisit = new LinkedList<>(Collections.nCopies(pointCount,Boolean.FALSE));

prevPoints = new ArrayList<>(Collections.nCopies(pointCount,0));

distance = new ArrayList<>(Collections.nCopies(pointCount,Integer.MAX\_VALUE));

edgesNumbers = new ArrayList<>(Collections.nCopies(pointCount,0));

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

ArrayList<Pair<Integer,Integer>> sublist = new ArrayList<Pair<Integer,Integer>>();

int edges = Integer.parseInt(scanner.next());

edgesNumbers.add(i,edges);

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

int fromPoint = Integer.parseInt(scanner.next());

int toPoint = Integer.parseInt(scanner.next());

int cost = Integer.parseInt(scanner.next());

sublist.add(j, new Pair<Integer, Integer>(toPoint, cost));

}

table.add(sublist);

}

}

}

**FormWorker.java**

import java.awt.BasicStroke;

import java.awt.BorderLayout;

import java.awt.Color;

import java.awt.Font;

import java.awt.Graphics2D;

import java.awt.event.WindowEvent;

import java.awt.event.WindowListener;

import java.awt.geom.Line2D;

import java.awt.image.BufferedImage;

import java.awt.image.ColorModel;

import java.awt.image.WritableRaster;

import java.io.File;

import java.io.FileNotFoundException;

import static java.lang.Math.cos;

import static java.lang.Math.sin;

import java.util.ArrayList;

import java.util.Map;

import javax.swing.JFileChooser;

import javax.swing.JOptionPane;

/\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

/\*\*

\*

\* @author Zhenya

\*/

public class FormWorker extends javax.swing.JFrame {

private Integer startPoint;

private DeikstraManager algoManager;

private GraphManager graph;

private BufferedImage background;

private BufferedImage curImage;

private double ax1,ax2,ay1,ay2;

private static final double angle = Math.PI/10;

private static final double len = 10;

/\*\*

\* Creates new form FormWorker

\*/

public FormWorker() {

addWindowListener(new WindowListener() {

public void windowOpened(WindowEvent event) { }

public void windowClosing(WindowEvent event) { }

public void windowActivated(WindowEvent event) {

jPanel1.getGraphics().drawImage(curImage, 0, 0,Color.BLACK, null);

}

public void windowClosed(WindowEvent event) { }

public void windowDeactivated(WindowEvent event) { }

public void windowDeiconified(WindowEvent event) { }

public void windowIconified(WindowEvent event) { }

});

initComponents();

}

/\*\*

\* This method is called from within the constructor to initialize the form.

\* WARNING: Do NOT modify this code. The content of this method is always

\* regenerated by the Form Editor.

\*/

@SuppressWarnings("unchecked")

// <editor-fold defaultstate="collapsed" desc="Generated Code">

private void initComponents() {

jPopupMenu1 = new javax.swing.JPopupMenu();

jButton3 = new javax.swing.JButton();

jPanel1 = new javax.swing.JPanel();

jLabel1 = new javax.swing.JLabel();

jLabel2 = new javax.swing.JLabel();

jButton1 = new javax.swing.JButton();

jLabel5 = new javax.swing.JLabel();

jScrollPane1 = new javax.swing.JScrollPane();

jTextPane1 = new javax.swing.JTextPane();

jButton4 = new javax.swing.JButton();

jButton2 = new javax.swing.JButton();

textArea1 = new java.awt.TextArea();

jButton3.setText("jButton3");

setDefaultCloseOperation(javax.swing.WindowConstants.EXIT\_ON\_CLOSE);

setTitle(" Визуализация алгоритма Дейкстры");

setPreferredSize(new java.awt.Dimension(700, 600));

setResizable(false);

jPanel1.setBackground(new java.awt.Color(255, 255, 255));

jPanel1.setPreferredSize(new java.awt.Dimension(600, 400));

jPanel1.setRequestFocusEnabled(false);

javax.swing.GroupLayout jPanel1Layout = new javax.swing.GroupLayout(jPanel1);

jPanel1.setLayout(jPanel1Layout);

jPanel1Layout.setHorizontalGroup(

jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addGap(0, 500, Short.MAX\_VALUE)

);

jPanel1Layout.setVerticalGroup(

jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addGap(0, 299, Short.MAX\_VALUE)

);

jLabel2.setText("Путь к файлу");

jLabel2.setToolTipText("");

jButton1.setText("Загрузить");

jButton1.setActionCommand("Загрузить граф");

jButton1.addActionListener(new java.awt.event.ActionListener() {

public void actionPerformed(java.awt.event.ActionEvent evt) {

jButton1ActionPerformed(evt);

}

});

jLabel5.setText("Стартовая вершина");

jScrollPane1.setViewportView(jTextPane1);

jButton4.setText("Следующий шаг");

jButton4.addActionListener(new java.awt.event.ActionListener() {

public void actionPerformed(java.awt.event.ActionEvent evt) {

jButton4ActionPerformed(evt);

}

});

jButton2.setText("Задать");

jButton2.addActionListener(new java.awt.event.ActionListener() {

public void actionPerformed(java.awt.event.ActionEvent evt) {

jButton2ActionPerformed(evt);

}

});

javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane());

getContentPane().setLayout(layout);

layout.setHorizontalGroup(

layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addGroup(javax.swing.GroupLayout.Alignment.TRAILING, layout.createSequentialGroup()

.addContainerGap(321, Short.MAX\_VALUE)

.addComponent(jButton4)

.addGap(237, 237, 237))

.addGroup(javax.swing.GroupLayout.Alignment.TRAILING, layout.createSequentialGroup()

.addGap(18, 18, 18)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING, false)

.addGroup(layout.createSequentialGroup()

.addComponent(jLabel2)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

.addComponent(jLabel1))

.addComponent(jButton1)

.addComponent(jLabel5, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, Short.MAX\_VALUE)

.addComponent(jScrollPane1))

.addComponent(jButton2))

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED, javax.swing.GroupLayout.DEFAULT\_SIZE, Short.MAX\_VALUE)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING, false)

.addComponent(textArea1, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, Short.MAX\_VALUE)

.addComponent(jPanel1, javax.swing.GroupLayout.DEFAULT\_SIZE, 500, Short.MAX\_VALUE))

.addGap(33, 33, 33))

);

layout.setVerticalGroup(

layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addGroup(layout.createSequentialGroup()

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addGroup(layout.createSequentialGroup()

.addGap(49, 49, 49)

.addComponent(jLabel2)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

.addComponent(jButton1)

.addGap(26, 26, 26)

.addComponent(jLabel1)

.addGap(18, 18, 18)

.addComponent(jLabel5)

.addGap(4, 4, 4)

.addComponent(jScrollPane1, javax.swing.GroupLayout.PREFERRED\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

.addComponent(jButton2))

.addGroup(layout.createSequentialGroup()

.addGap(21, 21, 21)

.addComponent(jPanel1, javax.swing.GroupLayout.PREFERRED\_SIZE, 299, javax.swing.GroupLayout.PREFERRED\_SIZE)))

.addGap(31, 31, 31)

.addComponent(textArea1, javax.swing.GroupLayout.PREFERRED\_SIZE, 173, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

.addComponent(jButton4)

.addContainerGap(24, Short.MAX\_VALUE))

);

pack();

}// </editor-fold>

private static BufferedImage deepCopy(BufferedImage bi) {

ColorModel cm = bi.getColorModel();

boolean isAlphaPremultiplied = cm.isAlphaPremultiplied();

WritableRaster raster = bi.copyData(null);

return new BufferedImage(cm, raster, isAlphaPremultiplied, null);

}

private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {

// TODO add your handling code here:

File file = null;

JFileChooser fileopen = new JFileChooser();

int ret = fileopen.showDialog(null, "Открыть файл");

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

file = fileopen.getSelectedFile();

}

graph = new GraphManager(startPoint);

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

try{

graph.readFromFile(file);

}

catch(FileNotFoundException e){System.out.println(e);}

background = new BufferedImage(500,400,BufferedImage.TYPE\_INT\_RGB);

Graphics2D graphics = background.createGraphics();

graphics.setFont(new Font("Serif", Font.PLAIN, 20));

graphics.setColor(Color.green);

graphics.fillRect(0, 0, background.getWidth(), background.getHeight());

graphics.setColor(Color.BLACK);

print\_points(graphics,graph);

print\_edges(graphics,graph,background);

this.setLayout(new BorderLayout());

jPanel1.getGraphics().drawImage(background, 0, 0,Color.BLACK, null);

this.pack();

this.setVisible(true);

try{

Thread.sleep(1000);

}catch(InterruptedException e){}

algoManager = new DeikstraManager(graph, background,this,this.textArea1,this.jPanel1);

algoManager.start();

}

private void jButton4ActionPerformed(java.awt.event.ActionEvent evt) {

algoManager.resume();

if(!algoManager.isAlive()){

String resLine ="";

for (int i = 0; i < graph.distance.size(); i++) {

resLine +="Вершина "+i+": ";

if(graph.distance.get(i) == Integer.MAX\_VALUE) resLine += String.valueOf((char)8734)+"\n";

else resLine += graph.distance.get(i)+"\n";

}

JOptionPane.showMessageDialog(this, "Алгоритм завершил работу.\nМетки вершин:\n"+resLine);

textArea1.append("\nАлгоритм завершил работу.\nМетки вершин:\n"+resLine);

jTextPane1.setText(null);

jPanel1.removeAll();

textArea1.setText(null);

jPanel1.repaint();

curImage = null;

graph = null;

}

}

private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {

// TODO add your handling code here:

startPoint = Integer.parseInt(jTextPane1.getText());

}

public void showEdge(BufferedImage background,GraphManager graph,int from,int to){

BufferedImage copyImage = deepCopy(background);

Graphics2D copyGraphics = (Graphics2D)copyImage.getGraphics();

this.showPointCost(copyGraphics);

double angle = 360 / graph.table.size()\*Math.PI / 180;

int centerX = 250;

int centerY = 150;

int radius = 125;

int startX = 250, startY = 25;

double curX1, curY1, curX2, curY2;

double cur\_angle1, cur\_angle2;

cur\_angle1 = from \* angle;

curX1 = (startX - centerX)\*cos(cur\_angle1) - (startY - centerY)\*sin(cur\_angle1) + centerX;

curY1 = (startX - centerX)\*sin(cur\_angle1) + (startY - centerY)\*cos(cur\_angle1) + centerY;

cur\_angle2 = to\*angle;

curX2 = (startX - centerX)\*cos(cur\_angle2) - (startY - centerY)\*sin(cur\_angle2) + centerX;

curY2 = (startX - centerX)\*sin(cur\_angle2) + (startY - centerY)\*cos(cur\_angle2) + centerY;

copyGraphics.setColor(Color.red);

copyGraphics.setStroke(new BasicStroke(3));

copyGraphics.drawLine((int)curX1, (int)curY1, (int)curX2, (int)curY2);

jPanel1.getGraphics().drawImage(copyImage, 0, 0, null);

this.pack();

this.setVisible(true);

curImage = this.deepCopy(copyImage);

}

public void showPointCost(Graphics2D copyGraphics){

double angle = 360 / graph.table.size()\*Math.PI / 180;

ArrayList<Map<Integer,Integer>> a;

int centerX = 250;

int centerY = 150;

int radius = 125;

double cur\_angle = 0;

int startX = 250, startY = 25;

double curX, curY;

for (int i = 0; i < graph.table.size(); i++, cur\_angle += angle)

{

curX = (startX - centerX)\*cos(cur\_angle) - (startY - centerY)\*sin(cur\_angle) + centerX;

curY = (startX - centerX)\*sin(cur\_angle) + (startY - centerY)\*cos(cur\_angle) + centerY;

copyGraphics.setFont(new Font("Tahoma", Font.PLAIN, 15));

copyGraphics.setColor(Color.red);

if(graph.distance.get(i) == Integer.MAX\_VALUE) copyGraphics.drawString(String.valueOf((char)8734), (int)curX - 15, (int) curY - 15);

else copyGraphics.drawString(String.valueOf(graph.distance.get(i)), (int)curX - 15, (int) curY - 15);

copyGraphics.setColor(Color.black);

}

}

public void dropPoint(GraphManager graph,BufferedImage image,int pointNum){

double angle = 360 / graph.table.size()\*Math.PI / 180;

int centerX = 250;

int centerY = 150;

int radius = 125;

double cur\_angle = 0;

int startX = 250, startY = 25;

double curX, curY;

Graphics2D graphics = (Graphics2D)image.getGraphics();

cur\_angle = pointNum\*angle;

curX = (startX - centerX)\*cos(cur\_angle) - (startY - centerY)\*sin(cur\_angle) + centerX;

curY = (startX - centerX)\*sin(cur\_angle) + (startY - centerY)\*cos(cur\_angle) + centerY;

graphics.setColor(Color.red);

graphics.setStroke(new BasicStroke(3));

graphics.drawOval((int)curX - 15, (int)curY - 15, 30, 30);

this.setVisible(true);

}

private void print\_points(Graphics2D graphics,GraphManager graph){

double angle = 360 / graph.table.size()\*Math.PI / 180;

ArrayList<Map<Integer,Integer>> a;

int centerX = 250;

int centerY = 150;

int radius = 125;

double cur\_angle = 0;

int startX = 250, startY = 25;

double curX, curY;

for (int i = 0; i < graph.table.size(); i++, cur\_angle += angle)

{

curX = (startX - centerX)\*cos(cur\_angle) - (startY - centerY)\*sin(cur\_angle) + centerX;

curY = (startX - centerX)\*sin(cur\_angle) + (startY - centerY)\*cos(cur\_angle) + centerY;

graphics.drawOval((int)curX - 15, (int)curY - 15, 30, 30);

graphics.setColor(Color.blue);

graphics.setFont(new Font("Tahoma", Font.PLAIN, 15));

graphics.drawString(String.valueOf(i), (int)curX, (int) curY);

graphics.setColor(Color.black);

}

}

private void print\_edges(Graphics2D graphics,GraphManager graph,BufferedImage image)

{

double angle = 360 / graph.table.size()\*Math.PI / 180;

int centerX = 250;

int centerY = 150;

int radius = 125;

int startX = 250, startY = 25;

double curX1, curY1, curX2, curY2;

double cur\_angle1, cur\_angle2;

for (int i = 0; i < graph.table.size(); i++)

{

for (int j = 0; j < graph.edgesNumbers.get(i); j++)

{

cur\_angle1 = i \* angle;

curX1 = (startX - centerX)\*cos(cur\_angle1) - (startY - centerY)\*sin(cur\_angle1) + centerX;

curY1 = (startX - centerX)\*sin(cur\_angle1) + (startY - centerY)\*cos(cur\_angle1) + centerY;

cur\_angle2 = graph.table.get(i).get(j).getFirst()\*angle;

curX2 = (startX - centerX)\*cos(cur\_angle2) - (startY - centerY)\*sin(cur\_angle2) + centerX;

curY2 = (startX - centerX)\*sin(cur\_angle2) + (startY - centerY)\*cos(cur\_angle2) + centerY;

graphics.draw (new Line2D.Double (curX1,curY1,curX2,curY2));

arrHead (curX1,curY1,curX2,curY2);

graphics.draw (new Line2D.Double (curX2,curY2,ax1,ay1));

graphics.draw (new Line2D.Double (curX2,curY2,ax2,ay2));

graphics.drawString(String.valueOf(graph.table.get(i).get(j).getSecond()), (int)(curX1 + curX2)/2, (int)(curY1 + curY2)/2);

}

}

}

private void arrHead (double x1, double y1, double x2, double y2)

{ double c,a,beta,theta,phi;

c = Math.sqrt ((x2-x1)\*(x2-x1) + (y2-y1)\*(y2-y1));

if (Math.abs(x2-x1) < 1e-6)

if (y2<y1) theta = Math.PI/2;

else theta = - Math.PI/2;

else

{ if (x2>x1)

theta = Math.atan ((y1-y2)/(x2-x1)) ;

else

theta = Math.atan ((y1-y2)/(x1-x2));

}

a = Math.sqrt (len\*len + c\*c - 2\*len\*c\*Math.cos(angle));

beta = Math.asin (len\*Math.sin(angle)/a);

phi = theta - beta;

ay1 = y1 - a \* Math.sin(phi); // coordinates of arrowhead endpoint

if (x2>x1)

ax1 = x1 + a \* Math.cos(phi);

else

ax1 = x1 - a \* Math.cos(phi);

phi = theta + beta; // second arrowhead endpoint

ay2 = y1 - a \* Math.sin(phi);

if (x2>x1)

ax2 = x1 + a \* Math.cos(phi);

else

ax2 = x1 - a \* Math.cos(phi);

}

/\*

/\*\*

\* @param args the command line arguments

\*/

// Variables declaration - do not modify

private javax.swing.JButton jButton1;

private javax.swing.JButton jButton2;

private javax.swing.JButton jButton3;

private javax.swing.JButton jButton4;

private javax.swing.JLabel jLabel1;

private javax.swing.JLabel jLabel2;

private javax.swing.JLabel jLabel5;

private javax.swing.JPanel jPanel1;

private javax.swing.JPopupMenu jPopupMenu1;

private javax.swing.JScrollPane jScrollPane1;

private javax.swing.JTextPane jTextPane1;

public java.awt.TextArea textArea1;

// End of variables declaration

}

**DeikstraManager.java**

import java.awt.Color;

import java.awt.TextArea;

import java.awt.image.BufferedImage;

import java.io.FileNotFoundException;

import java.util.ArrayList;

import java.util.Collections;

import javax.swing.JOptionPane;

import javax.swing.JPanel;

/\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

/\*\*

\*

\* @author Zhenya

\*/

public class DeikstraManager extends Thread{

private GraphManager graph;

public BufferedImage copyImage = null;

private FormWorker form;

private TextArea textArea;

private JPanel panel;

public DeikstraManager(GraphManager graph, BufferedImage copyImage, FormWorker form,TextArea area,JPanel panel) {

this.graph = graph;

this.copyImage = copyImage;

this.form = form;

textArea = area;

this.panel = panel;

}

public void run(){

graph.distance = new ArrayList<>(Collections.nCopies(graph.pointCount,Integer.MAX\_VALUE));

graph.distance.set(graph.startPoint, 0);

for (int i = 0; i < graph.pointCount; i++) {

int curPoint = -1;

for (int j = 0; j < graph.pointCount; j++) {

if(!graph.isVisit.get(j)&&(curPoint == -1 || graph.distance.get(j) < graph.distance.get(curPoint))){

curPoint = j;

}

}

if(graph.distance.get(curPoint) == Integer.MAX\_VALUE){

break;

}

graph.isVisit.set(curPoint, Boolean.TRUE);

form.dropPoint(graph, copyImage, curPoint);

for (int j = 0; j < graph.table.get(curPoint).size(); j++) {

this.suspend();

textArea.append("\nТекущая вершина = "+curPoint+"\n");

int to = graph.table.get(curPoint).get(j).getFirst();

int cost = graph.table.get(curPoint).get(j).getSecond();

form.showEdge(copyImage, graph, curPoint, to);

textArea.append("\nТекущее ребро = "+curPoint+"->"+to+" Стоимость ="+cost+"\n");

if(graph.distance.get(curPoint) + cost < graph.distance.get(to)){

textArea.append("Cтоимость ребра оптимальна, так как верно: ");

textArea.append(String.valueOf(graph.distance.get(curPoint)) + " + " + String.valueOf(cost) + " < " + String.valueOf(graph.distance.get(to))+"\n");

graph.distance.set(to, graph.distance.get(curPoint) + cost);

graph.prevPoints.set(to,curPoint);

}

else{

textArea.append("Cтоимость ребра не оптимальна, так как неверно: ");

textArea.append(String.valueOf(graph.distance.get(curPoint)) + " + " + String.valueOf(cost) + " < " + String.valueOf(graph.distance.get(to))+"\n");

}

textArea.append("Метки вершин:\n");

for (int k = 0; k < graph.distance.size();k++)

{

textArea.append("Вершина["+String.valueOf(k)+"] = "+String.valueOf(graph.distance.get(k))+"\n");

}

}

}

ArrayList<Integer> path = new ArrayList<>();

for (int v = graph.endPoint; v != graph.startPoint; v = graph.prevPoints.get(v))

{

path.add(v);

}

path.add(graph.startPoint);

}

}

**Тестирование:**

Все исходные данные хранятся в файле “input.txt”

|  |  |  |
| --- | --- | --- |
| Стартовая вершина | Конечная вершина | Конечные метки |
| 0 | 10 6  3  0 1 3  0 3 4  0 5 1  3  1 2 3  1 3 3  1 5 5  2  2 3 5  2 4 6  0  2  4 3 2  4 5 1  0 |  |
| 1 | 10 6  3  0 1 3  0 3 4  0 5 1  3  1 2 3  1 3 3  1 5 5  2  2 3 5  2 4 6  0  2  4 3 2  4 5 1  0 |  |
| 0 | 1 2  1  0 1 5  0 |  |
| 0 | 5 5  2  0 1 2  0 4 1  1  1 2 1  1  2 3 3  1  3 4 1  0 |  |

Так же при тестировании возникла проблема:при закрытии формы, содержимое пропадало. Данная проблема была решена при помощи введения WindowListeners.

**Вывод:**

В ходе работы над визуализацией работы алгоритма Дейкстры, я не только познакомился с работой алгоритма, но и закрепил навыки построения программ с использованием графических библиотек Swing и AWT, так же научился работать с системой контроля версий Git. Проект можно считать выполненным успешно, так как удалось создать программу визуализации алгоритма в отведенный срок и соответствующую требованиям к разработке. Так же нужно отметить, что план разработки облегчил и структурировал написание программы.