Федеральное государственное автономное образовательное учреждение

«Санкт-Петербургский государственный электротехнический университет

«ЛЭТИ» им. В.И. Ульянова (Ленина)»

Отчет

по практическому заданию

Выполнили: Вихляев В.В.

Осипов В.Г.

Андреев Н.А.

Факультет: КТИ

Группа №3304

Преподаватель:

Фирсов М.А.

Подпись преподавателя

Санкт-Петербург

2015

**Задание**

Визуализация алгоритма Форда-Беллмана.

**Спецификация**

*Внешняя спецификация****.***

Входные данные: взвешенный ориентированный граф, выделенная вершина.

Промежуточные данные: картинка, соответствующая каждому шагу алгоритма.

Выходные данные: расстояния от выделенной вершины до остальных вершин в графе.

*Место и форма представления входных и выходных данных****.***

Граф:  
- представление графа в файле:  
[номер 1 вершины] [пробел] [номер 2 вершины] [пробел] [вес вершины],

Выходные данные содержатся в специально выделенной таблице, которая содержит расстояния от выделенной вершины до других, полученных на каждом шаге. Визуальное представление графа содержит выделенную вершину, расстояния до других вершин, ребра, входящие в путь, отделенные цветом от других. Вся эта информация находится в главном окне программы.

*Интерфейс*.

рис.1. Схема главного окна программы.

Меню

Рисунок графа

Таблица (1), содержащая все ребра графа

Таблица (2), содержащая расстояния от выделенной вершины до остальных (изначально неактивна)

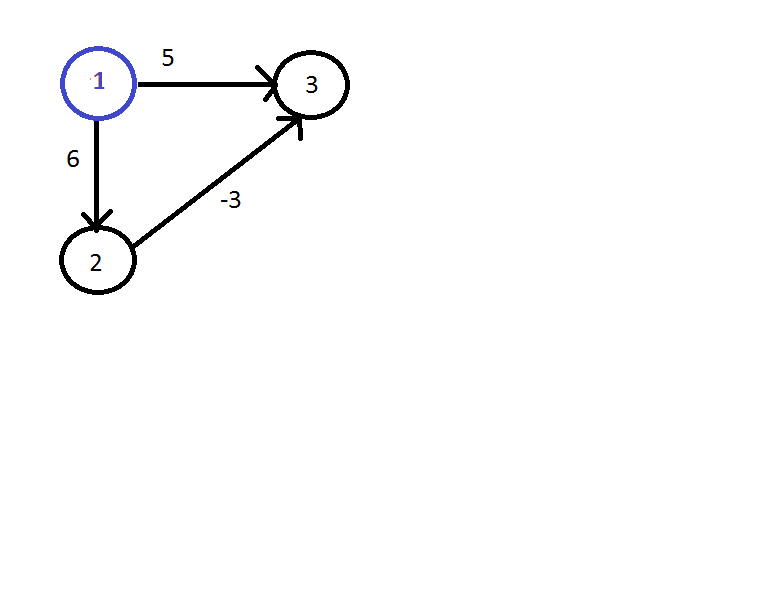
Кнопка «Показать алгоритм» («Сделать шаг»)

*Визуализация*. Выделенная вершина окрашивается в синий цвет.

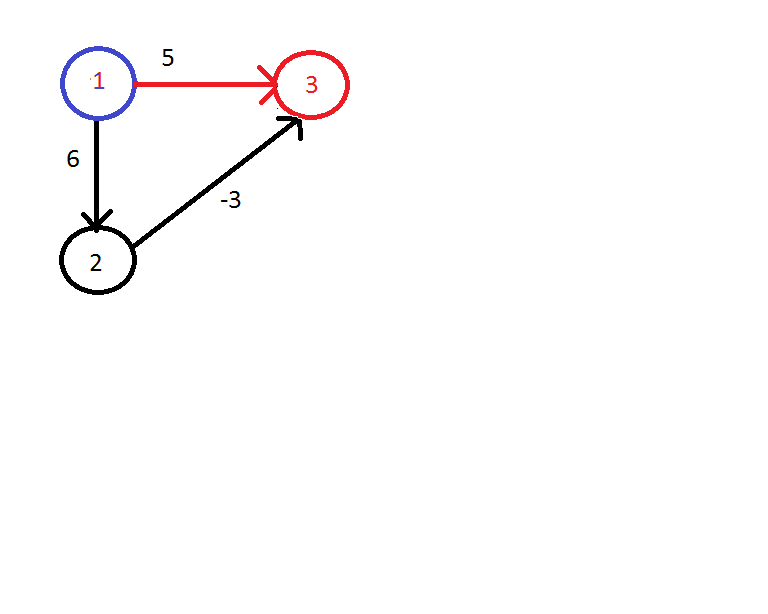
По нажатию кнопки «Сделать шаг»происходит следующее:  
- ослабляемое (релаксируемое) ребро и вершина, выделяются красным цветом,  
- ребра, входящие в путь, выделяются зеленым цветом,  
- в соответствии с изменениями на данном шаге изменяется таблица (2).

Рис. 2. Пример визуализации (первые несколько шагов).

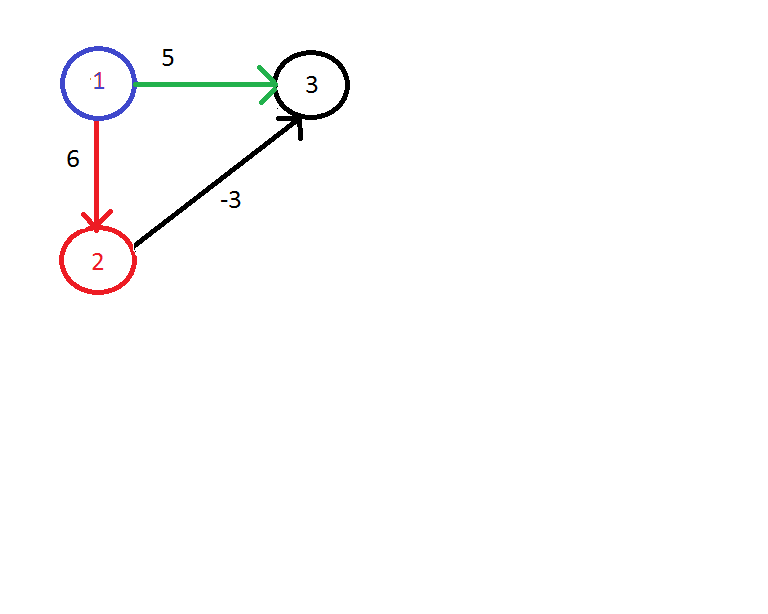
1 шаг.



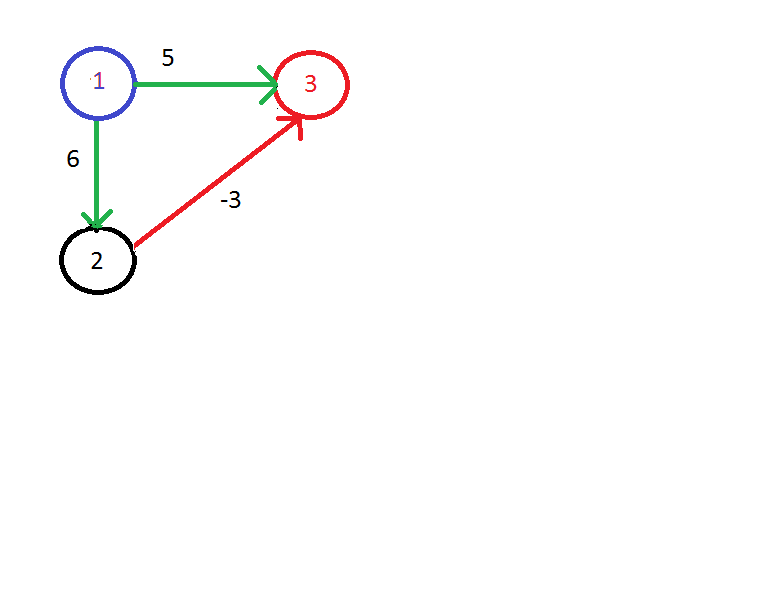
2 шаг.



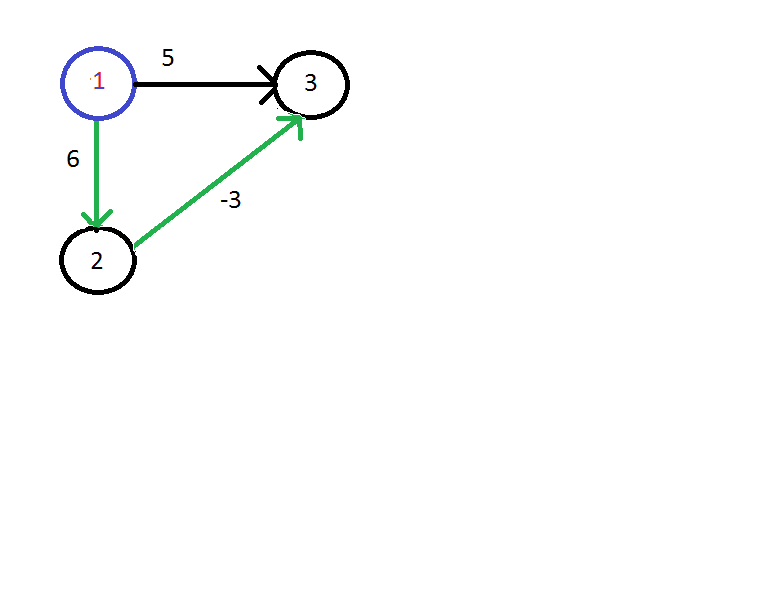
3 шаг.



4 шаг.



5 шаг.



*Внутренняя спецификация.*

Представление графа в программе, состоящего из N вершин и M ребер – список вершин, пронумерованных от 1 до N, и список ребер. Ребро – две вершины и вес этого ребра.  
Над графом определены базовые функции: добавление вершины, добавление ребра.

**Описание структуры данных:**

*Класс, реализующий граф и алгоритм.*

class Graph

{

static int INF = 1400;

static int MAX = 50;

int n; //число вершин

public int m; //число ребер

int k, u, v; //счетчика для алгоритма ФБ.

int dl[]; //массив из временных пометок

int consEdge; //номер рассматриваемого ребра при алгоритме ФБ

int s; //исходная вершина

boolean any; //для алгоритма ФБ

public Node e; //ссылка на начало списка ребер

public Point p[]; //массив из координат вершин

public Graph();

public Graph(int number);

public Graph(File file);

public void addVert();

public void addVert(int x, int y);

public void addEdge(int u, int v, int w);

public void draw\_dl();

public void drawGraph();

public int[] get\_table();

public String getEdge();

public int getN();

public String getEdges();

public boolean step();

public void Ford\_Bellman(int source);

};

*Класс - список ребер.*

class Node

{

int a, b, w; //a->b, weight = w

Node next;

public Node(int u, int v, int weigh);

public Node getNode(int i);

};

*Класс - точка*

class Point

{

int x,y;

public Point(int X, int Y);

public Point();

public void set(int newX, int newY);

public int getX();

public int getY();

};

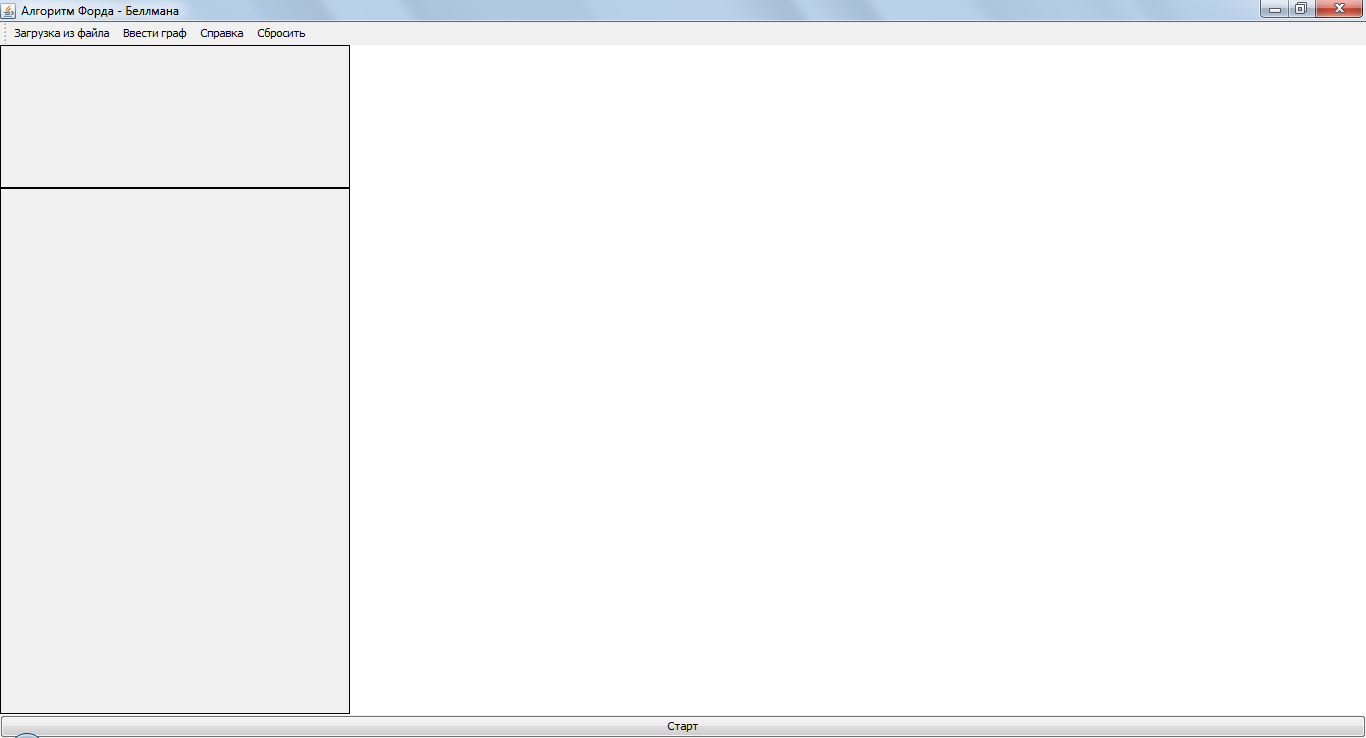
public static class myFrame - *Класс, реализующий пользовательский интерфейс.*

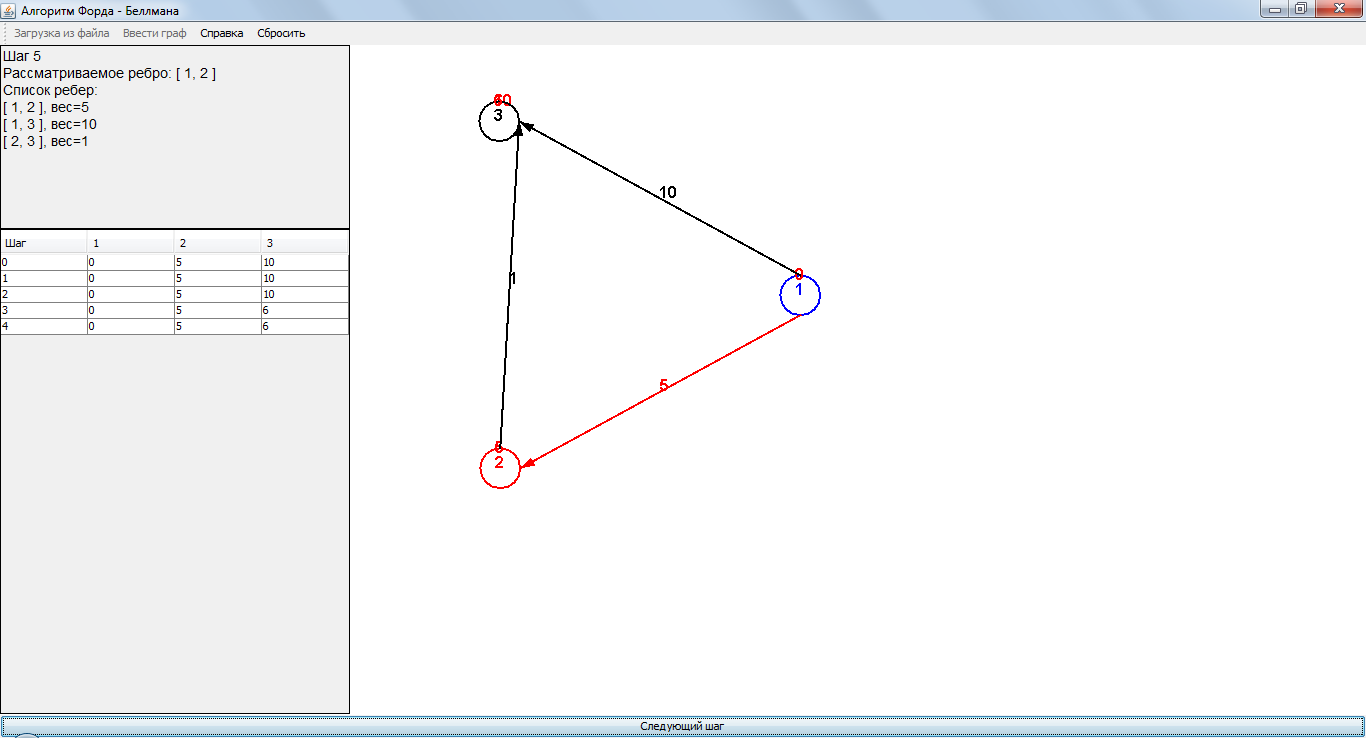
**План разработки**

|  |  |  |
| --- | --- | --- |
| Дата | Что должно быть готово | Ответственное лицо |
| 1 неделя | | |
| Среда | Внешняя спецификация, частично внутренняя, план разработки | Совместно. |
| Четверг | Репозиторий | Андреев. |
| Пятница | Прототип графического интерфейса. | Вихляев В. |
| Суббота | Пошаговая реализация алгоритма Ф-Б без графического интерфейса. | Осипов В. |
| 2 неделя | | |
| Понедельник | 1 версия приложения  Можно обойтись без:  - полной визуализации (допускается вывод только правильного ответа),  - корректности ввода (касается ввода из файла), - без рисования вручную /ввода из файла (должно быть сделано одно из двух)  - правильной реакции программы на неправильные действия (например, добавление вершины при выполнении алгоритма),  - красивого интерфейса (неполное меню, справка и т.д.). | Реализация интерфейса – Вихляев В.  Вывод графа на экран и ввод графа вручную – Андреев Н.  Реализация алгоритма – Осипов В. |
| Вторник | 1 версия приложения (если не был сделан) |
| Среда | 2 версия |
| Четверг | 2 версия + отчет, обновленный репозиторий |
| Пятница | Должно быть сдано/показано. |

**Пример работы программы**

*рис.3. Главное окно программы перед началом работы*



*рис.4.В процессе работы алгоритма*

**Заключение**

В процессе выполнения данного практического задания были получены базовые знания о языке Java, были освоены навыки программирования на языке Java, а также опыт создания программ в команде.

Результатом данной работы является приложение, наглядно показывающее алгоритм Форда-Беллмана. Во время разработки все исходные коды загружены на репозиторий.

**Приложение (код программы)**

Файл Graph.java

package practic;

import java.awt.BorderLayout;

import java.awt.Color;

import java.awt.Graphics;

import java.io.\*;

import java.util.Random;

import java.util.Scanner;

import java.util.concurrent.Semaphore;

import java.util.logging.Level;

import java.util.logging.Logger;

import javax.swing.\*;

class Vertex{

public int x, y; //координаты вершины

public int num;

public Vertex(int x, int y, int num){

this.x = x;

this.y = y;

this.num = num;

}

}

class Edge{

Vertex from;

Vertex to;

int weight;

Edge(Vertex from, Vertex to, int weight){

this.from = from;

this.to = to;

this.weight = weight;

}

}

public class Graph extends JPanel implements Runnable{

public int numV;

Semaphore sem;

public int numE;

public int numM;

int currentEdge;

JLabel doings;

public Thread graphThread;

public int r;

public Vertex v[];

public Edge e[];

public Edge MFT[];

public void run(){

doings.setText("");

repaint();

Kruskal();

}

protected void paintComponent(Graphics g){

super.paintComponent(g);

//настройка шрифта

g.setFont(g.getFont().deriveFont((float)r/2));

//отрисовка всех ребер графа

g.setColor(Color.BLACK);

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

g.drawLine(e[i].from.x+r/2, e[i].from.y+r/2,e[i].to.x+r/2,e[i].to.y+r/2);

}

//отрисовка ребер мод

g.setColor(Color.RED);

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

g.drawLine(MFT[i].from.x+r/2, MFT[i].from.y+r/2, MFT[i].to.x+r/2, MFT[i].to.y+r/2);

}

//отрисовка весов ребер

g.setColor(Color.BLUE);

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

String weight = Integer.toString(e[i].weight);

g.drawString(weight, (e[i].from.x + e[i].to.x +r)/2, (e[i].from.y + e[i].to.y +r)/2);

}

//отрисовка вершин

g.setColor(Color.CYAN);

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

g.fillOval(v[i].x,v[i].y, r, r);

}

if(currentEdge!=-1){

g.drawLine(e[currentEdge].from.x + r/2, e[currentEdge].from.y + r/2, e[currentEdge].to.x + r/2, e[currentEdge].to.y + r/2);

}

//отрисовка номеров вершин

g.setColor(Color.BLACK);

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

g.drawString(Integer.toString(i), v[i].x + 3\*r/8, v[i].y + 2\*r/3);

}

}

private void bothConstr(){

currentEdge = -1;

doings = new JLabel(""); //создание метки под вывод последовательности действий

graphThread = new Thread(this);

this.setLayout(new BorderLayout());

this.add(doings,BorderLayout.NORTH);

double angle = 6.2831853/numV; //вычиление расположения вершин

double nextangle = 0;

int R = 200;

r = (int)(6.28\*R)/(4\*numV);

if(r>50)r = 50;

if (r < 20){

//добавить скролл бар

r = 20;

}

int CentrX=250, CentrY=250;

int x, y;

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

x = (int) (CentrX + Math.cos(nextangle)\*R);

y = (int) (CentrY + Math.sin(nextangle)\*R);

nextangle += angle;

v[i] = new Vertex(x, y, i);

}

}

public Graph(int numvert, int proc, Semaphore sem){

//инициализация графа по заданым параметрам

int max = numvert\*(numvert-1)/2; //максимально возможное число верин в графе

int must = (int) (proc\*numvert\*(numvert-1)/100)/2; //необходимое нам число вершин

int counter = 0; //счетчик для массива вершин

numV = numvert; //инициализация полей графа

numE = must;

e = new Edge[must]; //создание массивов под вершины, ребра и мод

v = new Vertex[numvert];

MFT = new Edge[numV-1];

this.sem = sem;

bothConstr();

Random rand = new Random(System.currentTimeMillis()); //рандомные значения

Random weightrand = new Random(System.currentTimeMillis());

for(int i = 0; i < numvert && must > 0; i++){ //идем по вершинам, от которых можно провести ребро

for(int j = i+1; j < numvert && must > 0; j++){ //к которым можно его провести

if(rand.nextInt(max)< must){ //ставим ребро между i и j с вероятностью must/max

must--; //если ставим, уменьшаем оставшееся число ребер

e[counter] = new Edge(v[i],v[j],weightrand.nextInt(max)+1); //ставим ребро

counter++; //увеличиваем счетчик массива

}

max--; //уменьшаем максимально возможное оставшееся число ребер

}

}

graphThread.start();

}

public Graph(File file, Semaphore sem){

Scanner in;

try {

in = new Scanner(file);

this.sem = sem;

doings = new JLabel("");

numV = in.nextInt();

MFT = new Edge[numV-1];

v = new Vertex[numV];

numE = in.nextInt();

bothConstr();

e = new Edge[numE];

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

int from = in.nextInt();

int to = in.nextInt();

int weight = in.nextInt();

e[i] = new Edge(v[from], v[to], weight);

}

in.close();

graphThread.start();

} catch (FileNotFoundException ex) {

Logger.getLogger(Graph.class.getName()).log(Level.SEVERE, null, ex);

}

}

private void sortEdges(){

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

for(int j = i+1; j<numE; j++){

if(e[i].weight >e[j].weight){

Edge help = e[i];

e[i]=e[j];

e[j]=help;

}

}

}

}

public void Kruskal(){

sortEdges();

String result = "Последовательность добавления: ";

doings.setText(result);

boolean unions[][] = new boolean[numV][numV];

int counter = 0;

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

unions[j][j] = true;

}

for(int i = 0; i<numE && (counter < numV-1); i++){

try {

sem.acquire();

} catch (InterruptedException e1) {

// TODO Auto-generated catch block

e1.printStackTrace();

}

currentEdge = i;

doings.setText(result);

this.repaint();

try {

graphThread.sleep(2000);

} catch (InterruptedException e1) {

// TODO Auto-generated catch block

e1.printStackTrace();

}

Edge help = e[i];

int from = e[i].from.num;

int to = e[i].to.num;

int h = -1;

int h2 = -1;

for(int j = 0; (j < from && h == -1); j++)

if(unions[from][j])

h=j;

for(int j = 0; (j<to &&h2==-1); j++)

if(unions[to][j])

h2=j;

if(h2!=-1)to = h2;

if(h!=-1)from = h;

if(to<from){

h = to;

to = from;

from = h;

}

boolean diffUnions = true;

for(int j = 0; j< numV && diffUnions; j++){

if((unions[j][from] == unions[j][to])&&(unions[j][from] == true)){

diffUnions = false;

}

}

if(diffUnions){

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

if(unions[j][to]){

unions[j][from] = true;

}

}

MFT[counter] = help;

counter++;

result+=help.from.num+"-> "+help.to.num+" ;";

doings.setText(result);

currentEdge = -1;

this.repaint();

numM = counter;

}

else{

doings.setText("ЦИКЛ!");

currentEdge = -1;

this.repaint();

}

}

}

}

Файл Demo.java

package practic;

import java.awt.\*;

import java.util.\*;

import java.util.concurrent.Semaphore;

import java.awt.event.\*;

import java.io.File;

import javax.swing.\*;

public class Demo {

Thread mainThread;

JLabel step;

static int count;

Semaphore sem;

JFrame mainWindow;

JButton next;

Graph gr;

Demo(){

mainWindow = new JFrame("Визуализация алгоритма Краскала"); //создается контейнер, содержащий окно

mainWindow.setSize(1000, 700); //задается его размер

mainWindow.setLayout(new BorderLayout()); //расположение объектов на нем

mainWindow.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE); //чтобы закрывалася на

mainWindow.setVisible(true); //становится видимым

mainThread = Thread.currentThread();

sem = new Semaphore(1); //создается семафор на 1 разрешение

try {

sem.acquire();

} catch (InterruptedException e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

gr = new Graph(5,100,sem); //создается граф

mainWindow.add(gr, BorderLayout.CENTER); //добавляется на окно

step = new JLabel("Нажмите кнопку \"Далее\" для следующего шага"); //метка, отображающая номер шага

next = new JButton("Далее"); //оздается кнопка

next.addActionListener(new ActionListener(){ //задается действие при нажатии на кнопку

public void actionPerformed(ActionEvent ae){

if(gr.graphThread.isAlive()){

count++; //увеличивается счетчик шагов

step.setText("Шаг №"+Integer.toString(count)); //записывается номер шага

sem.release();

}

else {

step.setText("Завершено!");

}

}

});

JMenuBar menubar = new JMenuBar();

JMenu menu = new JMenu("Граф");

JMenuItem load = new JMenuItem("Загрузить из файла");

JMenuItem generate = new JMenuItem("Генерация");

JMenuItem show = new JMenuItem("Показать ребра");

menu.add(show);

//pp.setLayout( new BorderLayout());

menu.add(load);

menu.add(generate);

menubar.add(menu);

mainWindow.setJMenuBar(menubar);

show.addActionListener(new ActionListener(){

public void actionPerformed(ActionEvent AE){

JFrame edges = new JFrame("Ребра графа");

edges.setSize(500, 500); //задается его размер

edges.setLayout(new BorderLayout()); //расположение объектов на нем

edges.setDefaultCloseOperation(JFrame.HIDE\_ON\_CLOSE); //чтобы закрывалася на

edges.setVisible(true);

JTextArea list = new JTextArea();

//JLabel list = new JLabel();

String res = new String();

for(int i = 0; i < gr.numE; i++){

res+=" ИЗ "+gr.e[i].from.num+" В "+gr.e[i].to.num+" весом "+gr.e[i].weight+"\n";

}

list.setText(res);

list.setEditable(false);

//list.setText(res);

edges.add(list);

JScrollPane scroll = new JScrollPane(list);

edges.add(scroll);

}

});

load.addActionListener(new ActionListener(){

public void actionPerformed(ActionEvent ae){

FileDialog fileDialog = new FileDialog(mainWindow, "Выбор файла", FileDialog.LOAD);

fileDialog.show();

if(fileDialog.getFile()!=null){

count = 0;

step.setText("Нажмите кнопку \"Далее\" для следующего шага");

File file = new File(fileDialog.getDirectory()+ fileDialog.getFile());

gr.graphThread.stop();

sem = new Semaphore(1);

try {

sem.acquire();

} catch (InterruptedException e) {}

mainWindow.remove(gr);

gr = new Graph(file, sem);

mainWindow.add(gr);

}

}

});

generate.addActionListener(new ActionListener(){

public void actionPerformed(ActionEvent ae){

String numv =JOptionPane.showInputDialog(mainWindow,"Введите число вершин:","Выбор числа вершин", JOptionPane.PLAIN\_MESSAGE);

String proc = JOptionPane.showInputDialog(mainWindow,"Введите % ребер от 0 до 100:","Выбор % ребер", JOptionPane.PLAIN\_MESSAGE);

if(numv!=null &&proc != null ){

count = 0;

step.setText("Нажмите кнопку \"Далее\" для следующего шага");

gr.graphThread.stop();

sem = new Semaphore(1);

try {

sem.acquire();

} catch (InterruptedException e) {}

mainWindow.remove(gr);

gr = new Graph(Integer.parseInt(numv),Integer.parseInt(proc), sem);

mainWindow.add(gr);

}

}

});

mainWindow.add(next, BorderLayout.SOUTH);

mainWindow.add(step, BorderLayout.NORTH); //добавляем ее в контейнер

}

public static void main(String[] args) {

// TODO Auto-generated method stub

SwingUtilities.invokeLater(new Runnable(){

public void run(){

new Demo();

}

});

}

}