**Федеральное Агентство по Образованию**

Государственное образовательное учреждение высшего профессионального образования

**Санкт-Петербургский Государственный Электротехнический Университет «ЛЭТИ» им. В.И.Ульянова (Ленина)**

**(СПБГЭТУ)**

Кафедра МОЭВМ

Практический проект

«Алгоритм ЯПД»

“Выполнено” “\_\_\_\_” \_\_\_\_\_\_\_\_\_\_\_

Подпись преподавателя \_\_\_\_\_\_\_\_\_\_

Выполнили: Андросов В.

Исмаилов Р.

Степанов К.

Факультет КТИ

Группа № 4383

Преподаватель: Фирсов М.А.

Содержание:

[Формулировка задания: 3](#_Toc455355433)

[Спецификация: 3](#_Toc455355434)

[Описание алгоритма: 3](#_Toc455355435)

[Псевдокод: 3](#_Toc455355436)

[Оценка: 4](#_Toc455355437)

[Пример выполнения: 5](#_Toc455355438)

[Интерфейс программы(набросок): 11](#_Toc455355439)

[Итоговый вариант интерфейса: 12](#_Toc455355440)

[План разработки: 14](#_Toc455355441)

[Тестирование 14](#_Toc455355442)

[Описание структур данных 18](#_Toc455355443)

[Описание пакетов 18](#_Toc455355444)

[Вывод 18](#_Toc455355445)

[Код программы: 19](#_Toc455355446)

# **Формулировка задания:**

Реализовать визуализацию алгоритма построения минимального остовного дерева заданного графа(ЯПД) на языке Java.

# **Спецификация:**

# **Описание алгоритма:**

На вход алгоритма подаётся связный неориентированный граф. Для каждого ребра задаётся его стоимость.

Сначала берётся произвольная вершина и находится ребро, инцидентное данной вершине и обладающее наименьшей стоимостью. Найденное ребро и соединяемые им две вершины образуют дерево. Затем, рассматриваются рёбра графа, один конец которых — уже принадлежащая дереву вершина, а другой — нет; из этих рёбер выбирается ребро наименьшей стоимости. Выбираемое на каждом шаге ребро присоединяется к дереву. Таким образом, при выполнении каждого шага алгоритма, высота формируемого дерева увеличивается на 1. Рост дерева происходит до тех пор, пока не будут исчерпаны все вершины исходного графа.

Результатом работы алгоритма является остовное дерево минимальной стоимости.

# **Псевдокод:**

Обозначения:

# **Оценка:**

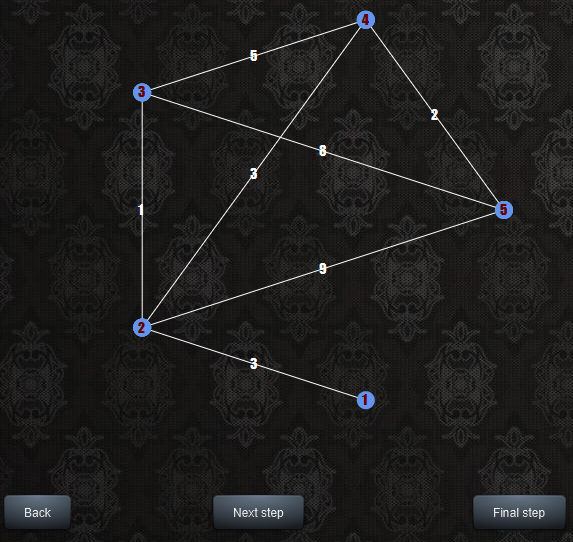
Асимптотика алгоритма зависит от способа хранения графа и способа хранения вершин, не входящих в дерево. Если приоритетная очередь Q реализована как обычный массив d, то Extract.min(Q) выполняется за O(n), а стоимость операции составляет O(1). Если Q представляет собой бинарную пирамиду, то стоимость Extract.min(Q) снижается до При использовании пирамид Фибоначчи, операция Extract.min(Q) выполняется за , а за O (1).

Выбранный способ хранения – массив матрицы смежности.

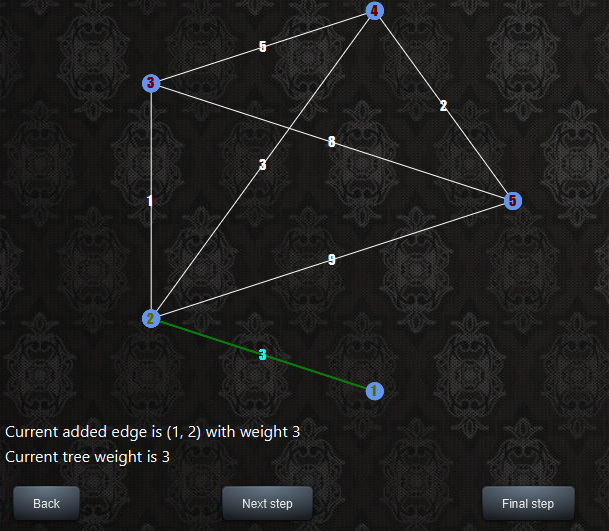
# **Пример выполнения:**

(выполнение производится нажатием кнопки Next)

1. Введенный граф



2) Добавлено ребро (1, 2)



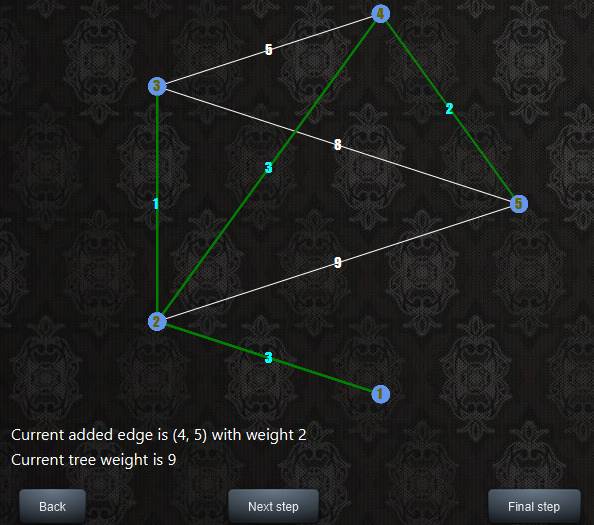
1. Добавлено ребро (2, 3)



1. Добавлено ребро (2, 4)



1. Добавлено ребро (4, 5)

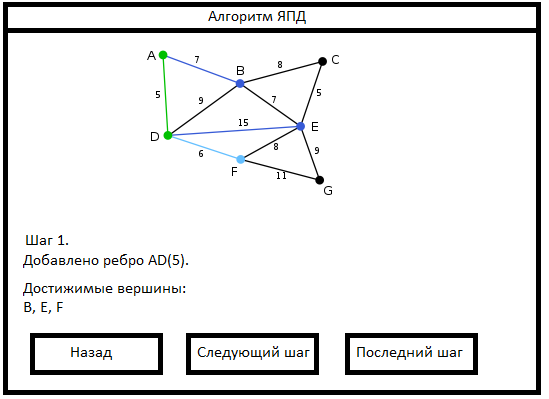


1. Все вершины графа включены, вес минимального остовного дерева, полученного алгоритмом ЯПД – 9.



Зеленым отмечены ребра графа, включенные в минимальное остовное дерево. Белым – ребра графа, не включенные в минимальное остовное дерево. Голубым отмечены веса ребер, включенных в минимальное остовное дерево, белым – веса ребер, не включенных в минимальное остовное дерево.

# **Интерфейс программы(набросок):**



Кнопка «Назад» ведет к диалоговому окну ввода графа.

Возможные варианты ввода:

* Подключаемый текстовый файл(.txt);
* Ввод в поле диалогового окна (поле текстового ввода, аналог Text box);
* Ввод в поле диалогового окна и из текстового файла производится в следующем формате:
* 1-я строка – целое число, обозначающее число вершин графа;
* Последующие строки: по 3 числа, обозначающие номера двух вершин, соединяемых ребром, и длину ребра соответственно.
* Ручное задание графа в графическом виде при помощи мыши (графическое поле с осями координат, двойной клик в поле для отметки вершины графа, соединить две вершины с зажатой левой кнопкой мыши для создания ребра между ними)

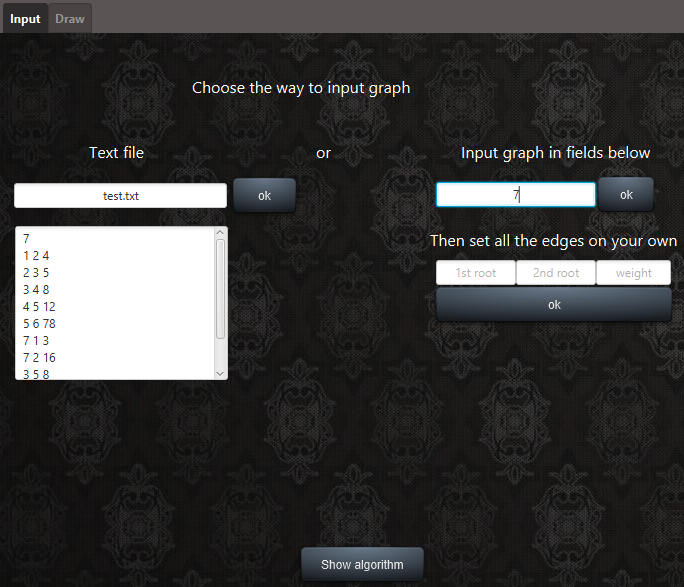
Кнопка «Следующий шаг» вызывает однократное выполнение тела цикла алгоритма. Кнопка остается активной до полного выполнения алгоритма.

Кнопка «Последний шаг» вызывает выполнение алгоритма полностью с отрисовкой конечного остовного дерева.

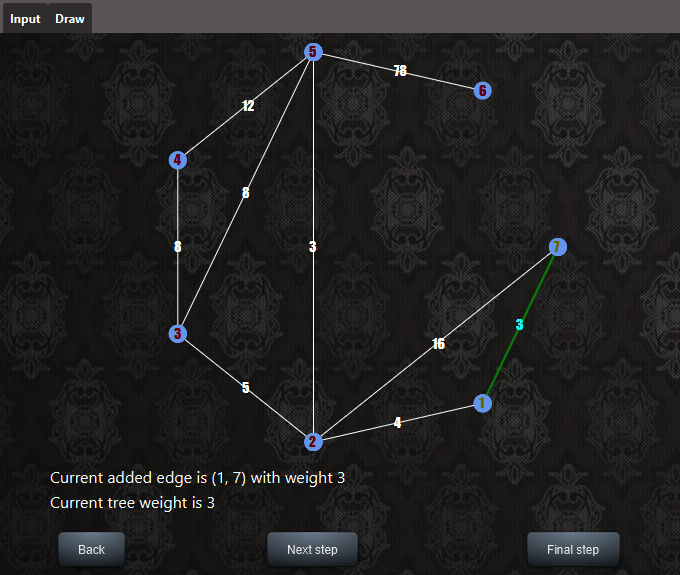
В поле «Шаг» отображается номер выполняемого шага алгоритма (без учета выбора начальной вершины), в поле «Достижимые вершины» отображаются вершины графа, для которых выполняется алгоритм на текущем шаге.

# **Итоговый вариант интерфейса:**

Ввод:



Отрисовка:



Распределение ролей в бригаде:

Андросов: реализация графического интерфейса для демонстрации графа и минимального остовного дерева.

Исмаилов: реализация инструментов демонстрации графов.

Степанов: реализация выполняемого алгоритма(ЯПД).

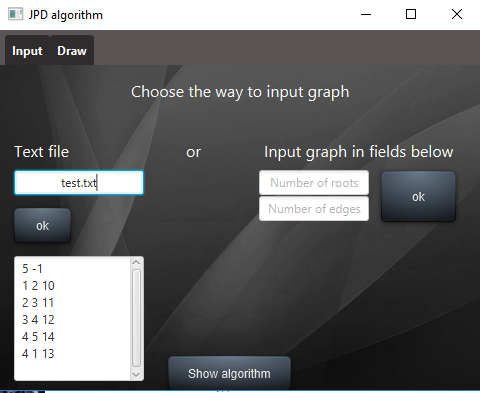
# **План разработки:**

|  |  |
| --- | --- |
| 27.06 | Итоговое согласование спецификации, создание прототипа GUI, разработка плана тестирования |
| 28.06 | Реализация рабочего алгоритма ЯПД |
| 28.06-29.06 | Реализация 1-й версии программы, тестирование, разработка документации. В 1-й версии программы необходимо реализовать ввод из файла, либо из поля диалогового окна (достаточно один из двух), а также отрисовку введенного графа и последнего шага алгоритма (готовое остовное дерево). |
| 30.06-01.07 | Реализация 2-й версии программы, тестирование. Во 2-й версии необходимо реализовать ввод из файла и из файла диалогового окна (оба варианта), а также отрисовку пошагового выполнения алгоритма. |
| 03.07 | Реализация финальной версии программы, сдача программы и документации. В финальной версии необходимо реализовать ввод графа в графическом поле с помощью мыши. |

# **Тестирование**

Первая версия:

Содержимое файла text.txt:



5 -1

1 2 10

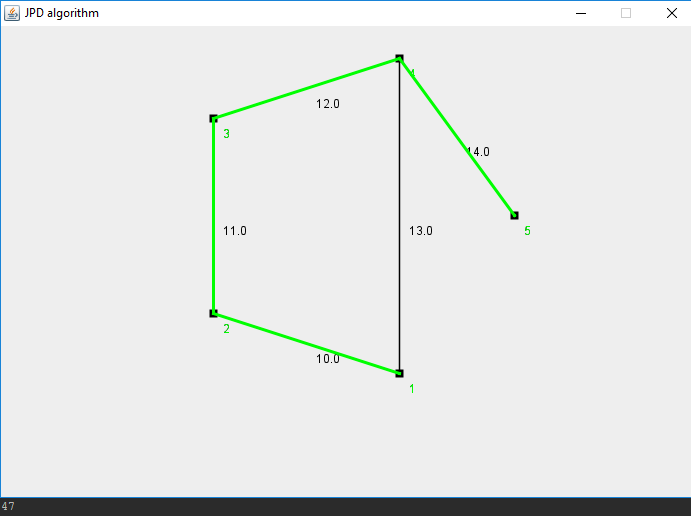
2 3 11

3 4 12

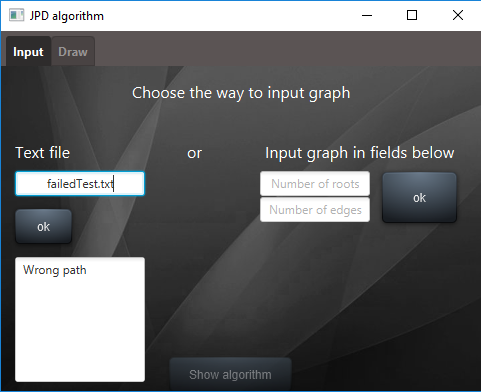
4 5 14

4 1 13

Граф и минимальное остовное дерево, построенные на данных из файла test.txt:

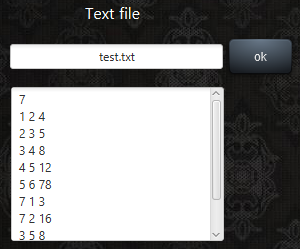


При некорректном вводе файла:



Вторая версия:

Содержимое файла test.txt:



7

1 2 4

2 3 5

3 4 8

4 5 12

5 6 78

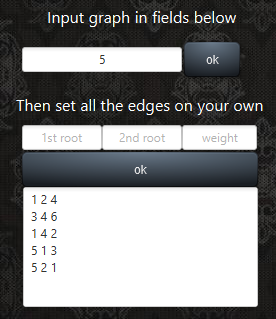
7 1 3

7 2 16

3 5 8

2 5 3

При нажатии кнопки «Последний шаг»:

Ручной ввод:

Количество вершин – 5

Ребра:

1 2 4

3 4 6

1 4 2

5 1 3

5 2 1

При трехкратном нажатии кнопки «Следующий шаг»:

# **Описание структур данных**

* public class GraphStruct {  
    
   public int first;  
   public int second;  
   public int weight;  
    
   public GraphStruct(int first, int second, int weight) {  
   this.first = first;  
   this.second = second;  
   this.weight = weight;  
   }  
  }

Основная структура программы, предназначена для хранения ребер в формате

<начальная вершина, конечная вершина, вес ребра>. Используется в совокупности со структурой данных List.

* int[][] graph

Матрица смежности графа;

* @Override  
  public void start(Stage primaryStage) throws Exception {  
   Parent root = FXMLLoader.load(getClass().getResource("view/sample.fxml"));  
   primaryStage.setTitle("DJP algorithm");  
   primaryStage.setScene(new Scene(root));  
   primaryStage.setResizable(false);  
   primaryStage.show();  
  }

Метод загрузки основного окна программы.

# **Описание пакетов**

Algorithm: пакет с реализацией алгоритма ЯПД;

GraphStruct: пакет со структурой данных для хранения ребер графа;

Main: пакет вызова метода main, загрузка основного окна программы;

Controller: пакет управления интерфейсом программы и отрисовки графа.

# **Вывод**

В данной работе представлена визуализация алгоритма ЯПД на графах. В работе использовался язык программирования Java, среда разработки Intellij IDEA, а также дополнительная библиотека JavaFX и модификация среды разработки Scene Builder.

Исходя из тестов, визуализация работы алгоритма достаточно корректна и наглядна.

# **Код программы:**

Содержимое файла Algorithm.java:

|  |
| --- |
| package sample;  import java.util.ArrayList; import java.util.List; import java.util.Vector;  /\*\*  \* Created by nphl on 01.07.2016.  \*/ public class Algorithm {   private int min(int a, int b) {  if (a < b) return a;  return b;  }   private void fill(int[] toFill, int value) {  for (int i = 0; i < toFill.length; i++) toFill[i] = value;  }   private void fill2(int[][] tof, int size, int value) {  for (int i = 0; i < size; i++) {  for (int j = 0; j < size; j++) tof[i][j] = value;  }  }   public int INF = Integer.MAX\_VALUE / 2; // "Бесконечность"  public int mstPrim(int[][] graph, int vNum, List<GraphStruct> toDraw) {   boolean[] used = new boolean[vNum]; // массив пометок  int[][] ddist = new int[vNum][3];  for (int i = 0; i < vNum; i++) ddist[i][0] = 0;  int[] dist = new int[vNum]; // массив расстояния. dist[v] = вес\_ребра(MST, v)  fill(dist, INF); // устанаавливаем расстояние до всех вершин INF  dist[0] = 0; // для начальной вершины положим 0  ddist[0][0] = 0;  for (; ; ) {  int v = -1;  for (int nv = 0; nv < vNum; nv++) // перебираем вершины  if (!used[nv] && dist[nv] < INF && (v == -1 || dist[v] > dist[nv]))  v = nv;// выбираем самую близкую непомеченную вершину   if (v == -1) break; // ближайшая вершина не найдена  used[v] = true; // помечаем ее  for (int nv = 0; nv < vNum; nv++) {  if (!used[nv] && graph[v][nv] < INF) { // для всех непомеченных смежных  dist[nv] = min(dist[nv], graph[v][nv]); // улучшаем оценку расстояния (релаксация)  if (graph[v][nv] == dist[nv]) {  ddist[nv][0] = dist[nv];  ddist[nv][1] = v;  ddist[nv][2] = nv;  }  }  }   }   for (int i = 1; i < vNum; i++) {  if (ddist[i][0] == 0) break;  toDraw.add(new GraphStruct(ddist[i][1]+1,ddist[i][2]+1,ddist[i][0]));  }  int ret = 0; // вес MST  for (int v = 0; v < vNum; v++)  ret += dist[v];  return ret;  } } |

Содержимое файла GraphStruct.java:

|  |
| --- |
| package sample;  /\*\*  \* Created by rossh on 03.07.2016.  \*/ public class GraphStruct {   public int first;  public int second;  public int weight;   public GraphStruct(int first, int second, int weight) {  this.first = first;  this.second = second;  this.weight = weight;   } } |

Содержимое файла Main.java:

|  |
| --- |
| package sample;  import javafx.application.Application; import javafx.fxml.FXMLLoader; import javafx.scene.Parent; import javafx.scene.Scene; import javafx.stage.Stage;  public class Main extends Application {   @Override  public void start(Stage primaryStage) throws Exception {  Parent root = FXMLLoader.load(getClass().getResource("view/sample.fxml"));  primaryStage.setTitle("DJP algorithm");  primaryStage.setScene(new Scene(root));  primaryStage.setResizable(false);  primaryStage.show();  }   public static void main(String[] args) {  launch(args);  } } |

Содержимое файла Controller.java:

|  |
| --- |
| package sample;  import javafx.fxml.FXML; import javafx.fxml.Initializable; import javafx.scene.control.\*; import javafx.scene.control.TextArea; import javafx.scene.control.TextField; import javafx.scene.control.Label; import javafx.scene.layout.Pane; import javafx.scene.paint.Color; import javafx.scene.shape.Circle; import javafx.scene.shape.Line;  import java.io.DataInputStream; import java.io.FileInputStream; import java.io.IOException; import java.net.URL; import java.util.\*;  public class Controller implements Initializable {   @FXML  TextField fileinptext;  @FXML  TextField numrootstext;  @FXML  TextField numribestext;  @FXML  TextField firstroottext;  @FXML  TextField secondroottext;  @FXML  TextField weighttext;   @FXML  Tab drawPage = new Tab();  @FXML  Tab inputPage = new Tab();  @FXML  TabPane mainTabPane = new TabPane();   @FXML  TextArea  filefield;  @FXML  TextArea  hinputfield;   @FXML  Button drawButton;  @FXML  Button okribebutton;  @FXML  Button finaldraw;  @FXML  Button backdraw;  @FXML  Button nextdraw;    @FXML  Label hinputlabel;  @FXML  Label drawLbl1;   @FXML  Label drawLbl2;   @FXML  Circle  baseDrawCircle;   @FXML  private Pane drawPane;   int[] inpInt = null;  int[] readyArr = null;  int size = 0;  int handInputTemp = 1;  String handInputTempStr = new String();  private int resultTree=0;  private Vector<GraphStruct> abc = new Vector<>();  private List<GraphStruct> tree = new ArrayList<>();   @FXML  public void enterPressed() {  fileOk();  }  @FXML  public void edgeEnterPressed() { okRBPressed(); }  @FXML  public void numOfRootsEnterPressed() { okRRPressed();}  @FXML  public void finalDrawPressed() {  drawTree();  drawLbl1.setVisible(false);  drawLbl2.setText("Minimum spanning tree weight is " + resultTree);  }    @FXML  public void backDrawPressed() {//TODO: clean up the graph picture by the button click  tree = null;  tree = new ArrayList<>();  abc = null;  abc = new Vector<>();  drawPage.setDisable(true);    SingleSelectionModel<Tab> selectionModel = mainTabPane.getSelectionModel();   if(selectionModel.getSelectedIndex() == 0){  selectionModel.select(1);  } else {  selectionModel.select(0);  }  }  int nextShowIndex = 1;  @FXML  public void nextDrawPressed(){    List CircList = new ArrayList<>(); //точки  List LblList = new ArrayList<>(); //лейблы  List LineList = new ArrayList<>(); // линии  Label V1, V2, V3;  Line edge;  double x1,x2,y1,y2;  int shiftx = 0, shifty=0;  int currWeight = 0;  for (int i = 0; ((i < tree.size()) && (i < nextShowIndex)); i++) {  currWeight += tree.get(i).weight;  x1 = baseDrawCircle.getRadius() \* Math.cos(2 \* Math.PI / size\*tree.get(i).first) + baseDrawCircle.getCenterX();  y1 = baseDrawCircle.getRadius() \* Math.sin(2 \* Math.PI / size \* tree.get(i).first) + baseDrawCircle.getCenterY();  x2 = baseDrawCircle.getRadius() \* Math.cos(2 \* Math.PI / size \* tree.get(i).second) + baseDrawCircle.getCenterX();  y2 = baseDrawCircle.getRadius() \* Math.sin(2 \* Math.PI / size \* tree.get(i).second) + baseDrawCircle.getCenterY();  // CircList.add(new Circle(x1, y1, 7, Color.GRAY));  // CircList.add(new Circle(x2, y2, 7, Color.GRAY));  // LblList.add(new Label(" " + i));  edge = new Line(x1, y1, x2, y2);  edge.setStrokeWidth(2);  edge.setStroke(Color.GREEN);  LineList.add(edge);  drawPane.getChildren().addAll(edge, new Circle(x1, y1, 9, Color.CORNFLOWERBLUE), new Circle(x2, y2, 9, Color.CORNFLOWERBLUE));  x1 += -4;  y1 += -9;  x2 += -4;  y2 += -9; //рисование имен вершин  V1 = new Label("" + tree.get(i).first);  V1.setLayoutX(x1);  V1.setLayoutY(y1);  V2 = new Label("" + tree.get(i).second);  V2.setLayoutX(x2);  V2.setLayoutY(y2);  V3 = new Label("" + tree.get(i).weight);  V3.setLayoutX((x1 + x2)/2 + shiftx);  V3.setLayoutY((y1 + y2)/2 + shifty);  V1.setStyle("-fx-font-weight: bold; -fx-font-size: 14px; -fx-text-fill: #666600; -fx-font-family: \"Impact\";");  V2.setStyle("-fx-font-weight: bold; -fx-font-size: 14px; -fx-text-fill: #666600; -fx-font-family: \"Impact\";");  V3.setStyle("-fx-font-weight: bold; -fx-font-size: 14px; -fx-text-fill: #00FFFF; -fx-font-family: \"Impact\";");  drawPane.getChildren().addAll(V1, V2, V3);  }  if(nextShowIndex>=size){  drawLbl1.setVisible(false);  drawLbl2.setText("Minimum spanning tree weight is " + resultTree);  }  else{  drawLbl1.setText("Current added edge is (" + tree.get(nextShowIndex-1).first + ", " + tree.get(nextShowIndex-1).second +  ") with weight " + tree.get(nextShowIndex-1).weight);  drawLbl2.setText("Current tree weight is " + currWeight);  }  nextShowIndex++;  }   @FXML  public void okRRPressed() {  if(numrootstext.getText().trim().isEmpty()) return;  size = Integer.parseInt(numrootstext.getText());  inpInt = new int[size \* (size) + 4];  inpInt[0] = size;   firstroottext.setVisible(true);  secondroottext.setVisible(true);  weighttext.setVisible(true);  okribebutton.setVisible(true);  hinputlabel.setVisible(true);  }   @FXML  public void okRBPressed() {  if(firstroottext.getText().trim().isEmpty()||secondroottext.getText().trim().isEmpty()||weighttext.getText().trim().isEmpty()) return;  inpInt[handInputTemp] = Integer.parseInt(firstroottext.getText());  inpInt[handInputTemp + 1] = Integer.parseInt(secondroottext.getText());  inpInt[handInputTemp + 2] = Integer.parseInt(weighttext.getText());  handInputTemp += 3;  handInputTempStr += firstroottext.getText() + " " + secondroottext.getText() + " " + weighttext.getText() + "\n";  hinputfield.setVisible(true);  hinputfield.setText(handInputTempStr);  firstroottext.setText("");  secondroottext.setText("");  weighttext.setText("");  drawButton.setDisable(false);  }   private void resizeInput() {  int j;  for (j = 0; j < inpInt.length; j++) if (inpInt[j] == 0) break;  readyArr = new int[j];  for (int i = 0; i < j; i++) readyArr[i] = inpInt[i];   GraphStruct temp;  for (int i = 0; i < (readyArr.length - 1) / 3; i++) {  temp = new GraphStruct(readyArr[i \* 3 + 1], readyArr[i \* 3 + 2], readyArr[i \* 3 + 3]);  abc.add(temp);  }  }   @FXML  public void drawPressed() {   resizeInput();  SingleSelectionModel<Tab> selectionModel = mainTabPane.getSelectionModel();   if(selectionModel.getSelectedIndex() == 0){  selectionModel.select(1);  } else {  selectionModel.select(0);  }  drawPage.setDisable(false);  drawGraph();  Algorithm prim = new Algorithm();  int[][] adjMatrix = new int[size][size];  for(int i=0; i<size;i++){  for(int j=0; j<size;j++) adjMatrix[i][j] = prim.INF;  }  for(int i=0; i<abc.size();i++){  adjMatrix[abc.get(i).first-1][abc.get(i).second-1] = (int)abc.get(i).weight;  adjMatrix[abc.get(i).second-1][abc.get(i).first-1] = (int)abc.get(i).weight;  }  resultTree = prim.mstPrim(adjMatrix,size,tree);  sortTree();  }   @FXML  public void fileOk() {  int i = 0;  String buff, buff2 = new String();  DataInputStream input = null;  try (DataInputStream streambuff = new DataInputStream(new FileInputStream(fileinptext.getText()))) {  size = streambuff.available();  } catch (IOException ex2) {  filefield.setText("Wrong path");  filefield.setVisible(true);  return;  }  inpInt = new int[size];  try {  input = new DataInputStream(new FileInputStream(fileinptext.getText()));   while ((buff = input.readLine()) != null) {  StringTokenizer stoken = new StringTokenizer(buff);  while (stoken.hasMoreTokens()) {  int ibuff = Integer.parseInt(stoken.nextToken());  inpInt[i] = ibuff;  buff2 += Integer.toString(ibuff) + " ";  if (i % 3 == 0) buff2 += "\n";  i++;  }  }  } catch (IOException ex) {   }  size = inpInt[0];  filefield.setText(buff2);  filefield.setVisible(true);  //drawPage.setDisable(false);  drawButton.setDisable(false);  }   private void drawGraph(){   List CircList = new ArrayList<>(); //точки  List LblList = new ArrayList<>(); //лейблы  List LineList = new ArrayList<>(); // линии  Label V1, V2, V3;  Line edge;  double x1,x2,y1,y2;  int shiftx = 0, shifty=0;  for (int i = 0; i < abc.size(); i++) {  x1 = baseDrawCircle.getRadius() \* Math.cos(2 \* Math.PI / size\*abc.get(i).first) + baseDrawCircle.getCenterX();  y1 = baseDrawCircle.getRadius() \* Math.sin(2 \* Math.PI / size \* abc.get(i).first) + baseDrawCircle.getCenterY();  x2 = baseDrawCircle.getRadius() \* Math.cos(2 \* Math.PI / size \* abc.get(i).second) + baseDrawCircle.getCenterX();  y2 = baseDrawCircle.getRadius() \* Math.sin(2 \* Math.PI / size \* abc.get(i).second) + baseDrawCircle.getCenterY();  // CircList.add(new Circle(x1, y1, 7, Color.GRAY));  // CircList.add(new Circle(x2, y2, 7, Color.GRAY));  // LblList.add(new Label(" " + i));  edge = new Line(x1, y1, x2, y2);  edge.setStrokeWidth(1);  edge.setStroke(Color.WHITE);  LineList.add(edge);  drawPane.getChildren().addAll(edge, new Circle(x1, y1, 9, Color.CORNFLOWERBLUE), new Circle(x2, y2, 9, Color.CORNFLOWERBLUE));  x1 += -4;  y1 += -9;  x2 += -4;  y2 += -9; //рисование имен вершин  V1 = new Label("" + abc.get(i).first);  V1.setLayoutX(x1);  V1.setLayoutY(y1);  V2 = new Label("" + abc.get(i).second);  V2.setLayoutX(x2);  V2.setLayoutY(y2);  V3 = new Label("" + abc.get(i).weight);  V3.setLayoutX((x1 + x2)/2 + shiftx);  V3.setLayoutY((y1 + y2)/2 + shifty);  V1.setStyle("-fx-font-weight: bold; -fx-font-size: 14px; -fx-text-fill: #660000; -fx-font-family: \"Impact\";");  V2.setStyle("-fx-font-weight: bold; -fx-font-size: 14px; -fx-text-fill: #660000; -fx-font-family: \"Impact\";");  V3.setStyle("-fx-font-weight: bold; -fx-font-size: 14px; -fx-text-fill: #FFFFFF; -fx-font-family: \"Impact\";");  drawPane.getChildren().addAll(V1, V2, V3);  }    }  private void drawTree(){   List CircList = new ArrayList<>(); //точки  List LblList = new ArrayList<>(); //лейблы  List LineList = new ArrayList<>(); // линии  Label V1, V2, V3;  Line edge;  double x1,x2,y1,y2;  int shiftx = 0, shifty=0;  for (int i = 0; i < tree.size(); i++) {  x1 = baseDrawCircle.getRadius() \* Math.cos(2 \* Math.PI / size\*tree.get(i).first) + baseDrawCircle.getCenterX();  y1 = baseDrawCircle.getRadius() \* Math.sin(2 \* Math.PI / size \* tree.get(i).first) + baseDrawCircle.getCenterY();  x2 = baseDrawCircle.getRadius() \* Math.cos(2 \* Math.PI / size \* tree.get(i).second) + baseDrawCircle.getCenterX();  y2 = baseDrawCircle.getRadius() \* Math.sin(2 \* Math.PI / size \* tree.get(i).second) + baseDrawCircle.getCenterY();  // CircList.add(new Circle(x1, y1, 7, Color.GRAY));  // CircList.add(new Circle(x2, y2, 7, Color.GRAY));  // LblList.add(new Label(" " + i));  edge = new Line(x1, y1, x2, y2);  edge.setStrokeWidth(2);  edge.setStroke(Color.GREEN);  LineList.add(edge);  drawPane.getChildren().addAll(edge, new Circle(x1, y1, 9, Color.CORNFLOWERBLUE), new Circle(x2, y2, 9, Color.CORNFLOWERBLUE));  x1 += -4;  y1 += -9;  x2 += -4;  y2 += -9; //рисование имен вершин  V1 = new Label("" + tree.get(i).first);  V1.setLayoutX(x1);  V1.setLayoutY(y1);  V2 = new Label("" + tree.get(i).second);  V2.setLayoutX(x2);  V2.setLayoutY(y2);  V3 = new Label("" + tree.get(i).weight);  V3.setLayoutX((x1 + x2)/2 + shiftx);  V3.setLayoutY((y1 + y2)/2 + shifty);  V1.setStyle("-fx-font-weight: bold; -fx-font-size: 14px; -fx-text-fill: #666600; -fx-font-family: \"Impact\";");  V2.setStyle("-fx-font-weight: bold; -fx-font-size: 14px; -fx-text-fill: #666600; -fx-font-family: \"Impact\";");  V3.setStyle("-fx-font-weight: bold; -fx-font-size: 14px; -fx-text-fill: #00FFFF; -fx-font-family: \"Impact\";");  drawPane.getChildren().addAll(V1, V2, V3);  }  }   @Override  public void initialize(URL location, ResourceBundle res) {   drawPage.setDisable(true);  filefield.setVisible(false);  drawButton.setDisable(true);  hinputfield.setVisible(false);  okribebutton.setVisible(false);  firstroottext.setVisible(false);  secondroottext.setVisible(false);  weighttext.setVisible(false);  hinputlabel.setVisible(false);  drawLbl1.setText("");  drawLbl2.setText("");  }   private void sortTree(){  int buff = 0;  for(int i = 0;i < tree.size();i++){  if(tree.get(i).first>tree.get(i).second){  buff = tree.get(i).first;  tree.get(i).first = tree.get(i).second;  tree.get(i).second = buff;  }  }  boolean swapped = true;  int j = 0;  int tmp;  while (swapped) {  swapped = false;  j++;  for (int i = 0; i < tree.size() - j; i++) {  if (tree.get(i).first > tree.get(i+1).first) {  tmp = tree.get(i).first;  tree.get(i).first = tree.get(i+1).first;  tree.get(i+1).first = tmp;   tmp = tree.get(i).second;  tree.get(i).second = tree.get(i+1).second;  tree.get(i+1).second = tmp;   tmp = tree.get(i).weight;  tree.get(i).weight = tree.get(i+1).weight;  tree.get(i+1).weight = tmp;   swapped = true;  }  }  }  swapped = true; j=0;  while (swapped) {  swapped = false;  j++;  for (int i = 0; i < tree.size() - j; i++) {  if ((tree.get(i).weight > tree.get(i+1).weight)&&(tree.get(i).first==tree.get(i+1).first)) {  tmp = tree.get(i).first;  tree.get(i).first = tree.get(i+1).first;  tree.get(i+1).first = tmp;   tmp = tree.get(i).second;  tree.get(i).second = tree.get(i+1).second;  tree.get(i+1).second = tmp;   tmp = tree.get(i).weight;  tree.get(i).weight = tree.get(i+1).weight;  tree.get(i+1).weight = tmp;   swapped = true;  }  }  }  }   } |

Содержимое файла sample.fxml:

|  |
| --- |
| <?xml version="1.0" encoding="UTF-8"?>  <?import javafx.scene.shape.\*?> <?import javafx.scene.chart.\*?> <?import javafx.scene.text.\*?> <?import javafx.scene.control.\*?> <?import java.lang.\*?> <?import javafx.scene.layout.\*?> <?import javafx.geometry.Insets?> <?import javafx.scene.layout.GridPane?> <?import javafx.scene.control.Button?> <?import javafx.scene.control.Label?>  <AnchorPane maxHeight="-Infinity" maxWidth="-Infinity" minHeight="-Infinity" minWidth="-Infinity" prefHeight="600.0" prefWidth="700.0" stylesheets="@../gui.css" xmlns="http://javafx.com/javafx/8" xmlns:fx="http://javafx.com/fxml/1" fx:controller="sample.Controller">  <children>  <TabPane fx:id="mainTabPane" layoutX="150.0" layoutY="70.0" prefHeight="360.0" prefWidth="480.0" stylesheets="@../gui.css" tabClosingPolicy="UNAVAILABLE" AnchorPane.bottomAnchor="0.0" AnchorPane.leftAnchor="0.0" AnchorPane.rightAnchor="0.0" AnchorPane.topAnchor="0.0">  <tabs>  <Tab fx:id="inputPage" text="Input">  <content>  <AnchorPane maxHeight="-Infinity" maxWidth="-Infinity" minHeight="-Infinity" minWidth="-Infinity" prefHeight="600.0" prefWidth="700.0" stylesheets="@../gui.css">  <children>  <Label fx:id="lbl1" layoutX="194.0" layoutY="42.0" text="Choose the way to input graph" textFill="WHITE">  <font>  <Font size="16.0" />  </font>  </Label>  <Label layoutX="91.0" layoutY="107.0" prefHeight="17.0" prefWidth="63.0" text="Text file" textFill="WHITE">  <font>  <Font size="16.0" />  </font>  </Label>  <TextField fx:id="fileinptext" alignment="CENTER" layoutX="16.0" layoutY="150.0" minHeight="-Infinity" onAction="#enterPressed" prefHeight="25.0" prefWidth="213.0" promptText="input file path here" AnchorPane.bottomAnchor="425.0" AnchorPane.leftAnchor="16.0" AnchorPane.rightAnchor="471.0" AnchorPane.topAnchor="150.0" />  <Button fx:id="fileinpbutton" layoutX="235.0" layoutY="145.0" mnemonicParsing="false" onAction="#fileOk" prefHeight="35.0" prefWidth="63.0" text="ok" />  <TextField fx:id="firstroottext" alignment="CENTER" layoutX="438.0" layoutY="227.0" prefHeight="25.0" prefWidth="80.0" promptText="1st root" />  <TextField fx:id="secondroottext" alignment="CENTER" layoutX="518.0" layoutY="227.0" prefHeight="25.0" prefWidth="80.0" promptText="2nd root" />  <TextField fx:id="weighttext" alignment="CENTER" layoutX="598.0" layoutY="227.0" onAction="#edgeEnterPressed" prefHeight="25.0" prefWidth="75.0" promptText="weight" />  <Label fx:id="rootsandribesbutton" layoutX="463.0" layoutY="107.0" text="Input graph in fields below" textFill="WHITE">  <font>  <Font size="16.0" />  </font>  </Label>  <Button fx:id="okRR" layoutX="600.0" layoutY="144.0" mnemonicParsing="false" onAction="#okRRPressed" prefHeight="35.0" prefWidth="56.0" text="ok" />  <TextField fx:id="numrootstext" alignment="CENTER" layoutX="438.0" layoutY="149.0" minWidth="-Infinity" onAction="#numOfRootsEnterPressed" prefHeight="25.0" prefWidth="160.0" promptText="Number of roots" />  <Button fx:id="okribebutton" layoutX="438.0" layoutY="254.0" mnemonicParsing="false" onAction="#okRBPressed" prefHeight="35.0" prefWidth="236.0" text="ok" />  <Label layoutX="318.0" layoutY="107.0" text="or" textFill="WHITE">  <font>  <Font size="16.0" />  </font>  </Label>  <Label fx:id="hinputlabel" layoutX="432.0" layoutY="195.0" text="Then set all the edges on your own" textFill="WHITE">  <font>  <Font size="16.0" />  </font>  </Label>  <TextArea fx:id="filefield" layoutX="17.0" layoutY="193.0" prefHeight="154.0" prefWidth="213.0" />  <Button fx:id="drawButton" layoutX="303.0" layoutY="514.0" mnemonicParsing="false" onAction="#drawPressed" text="Show algorithm" />  <TextArea fx:id="hinputfield" layoutX="439.0" layoutY="289.0" prefHeight="120.0" prefWidth="235.0" />  </children>  </AnchorPane>  </content>  </Tab>  <Tab fx:id="drawPage" text="Draw">  <content>  <AnchorPane maxHeight="-Infinity" maxWidth="-Infinity" minHeight="-Infinity" minWidth="-Infinity" prefHeight="600.0" prefWidth="700.0" stylesheets="@../gui.css">  <children>  <Label fx:id="drawLbl1" layoutX="52.0" layoutY="432.0" text="Current step: " textFill="WHITE">  <font>  <Font size="16.0" />  </font>  </Label>  <Label fx:id="drawLbl2" layoutX="52.0" layoutY="457.0" text="Current edges:" textFill="WHITE">  <font>  <Font size="16.0" />  </font>  </Label>  <Pane fx:id="drawPane" layoutX="60.0" layoutY="14.0" prefHeight="400.0" prefWidth="600.0">  <children>  <Circle fx:id="baseDrawCircle" centerX="300.0" centerY="200.0" fill="DODGERBLUE" layoutX="600.0" layoutY="386.0" radius="200.0" stroke="BLACK" strokeType="INSIDE" />  </children></Pane>  <Button fx:id="backdraw" layoutX="60.0" layoutY="499.0" mnemonicParsing="false" onAction="#backDrawPressed" text="Back" />  <Button fx:id="nextdraw" layoutX="269.0" layoutY="499.0" mnemonicParsing="false" onAction="#nextDrawPressed" text="Next step" />  <Button fx:id="finaldraw" layoutX="529.0" layoutY="499.0" mnemonicParsing="false" onAction="#finalDrawPressed" text="Final step" />  </children>  </AnchorPane>  </content>  </Tab>  </tabs>  </TabPane>  <Circle fill="DODGERBLUE" radius="100.0" stroke="BLACK" strokeType="INSIDE" visible="false" />  </children> </AnchorPane> |

Содержимое файла gui.css:

|  |
| --- |
| .root {  -fx-background-image: url("backgr.png"); }  .tab-pane > .tab-header-area > .headers-region > .tab:top {  -fx-border-color: #2e2c2c;  -fx-border-radius: 5 5 0 0;  -fx-background-color: #2e2c2c;  -fx-pref-height: 30px;  -fx-background-insets: 0; } .tab-pane > .tab-header-area > .headers-region > .tab:selected > .tab-container > .tab-label {  -fx-alignment: CENTER;  -fx-text-fill: #2e2c2c;  -fx-font-weight: bold; } .tab-pane > .tab-header-area > .tab-header-background {  -fx-background-color: #5b5454;  -fx-border-color: #5b5454;  -fx-border-width: 0 0 1 0;  -fx-background-insets: 0;  -fx-pref-height: 45px; }  .tab-pane > .tab-header-area > .headers-region > .tab:selected > .tab-container > .tab-label {  -fx-alignment: CENTER;  -fx-text-fill: white;  -fx-font-weight: bold; }  .tab-pane > .tab-header-area > .headers-region > .tab > .tab-container > .tab-label {  -fx-alignment: CENTER;  -fx-text-fill: white;  -fx-font-weight: bold; } .button {  -fx-background-color: #090a0c,  linear-gradient(#38424b 0%, #1f2429 20%, #191d22 100%),  linear-gradient(#20262b, #191d22),  radial-gradient(center 50% 0%, radius 100%, rgba(114, 131, 148, 0.9), rgba(255, 255, 255, 0));  -fx-background-radius: 5, 4, 3, 5;  -fx-background-insets: 0, 1, 2, 0;  -fx-text-fill: white;  -fx-effect: dropshadow(three-pass-box, rgba(0, 0, 0, 0.6), 5, 0.0, 0, 1);  -fx-font-family: "Arial";  -fx-text-fill: linear-gradient(white, #d0d0d0);  -fx-font-size: 12px;  -fx-padding: 10 20 10 20; } |