**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра Математического обеспечения и применения ЭВМ**

отчет

**по лабораторной работе № 3**

**по дисциплине «Построение и Анализ Алгоритмов»**

Тема: Нахождение кратчайшего расстояния между двумя вершинами.

|  |  |  |
| --- | --- | --- |
| Студент гр. 6304 |  | Ковынев М.В. |
| Преподаватель |  | Балтрашевич В.Э. |

Санкт-Петербург

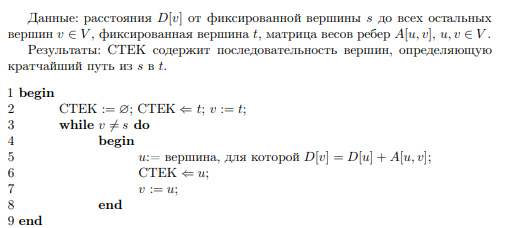
2018

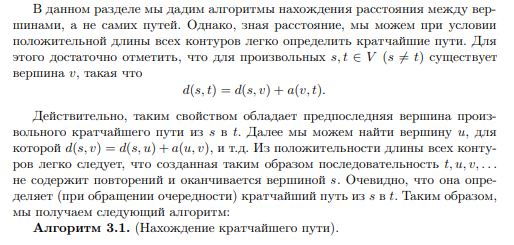
# **Цель работы**

Реализовать алгоритм нахождение кратчайшего пути между любыми 2 вершинами ориентированного взвешенного графа.

# **Ход работы**

1. Реализуем функцию, которая обходит матрицу смежности и находит кратчайшее расстояние между любыми 2 вершинами ориентированного взвешенного графа.

Приведем описание алгоритма из книги «Комбинаторика для программистов» - В. Липский.



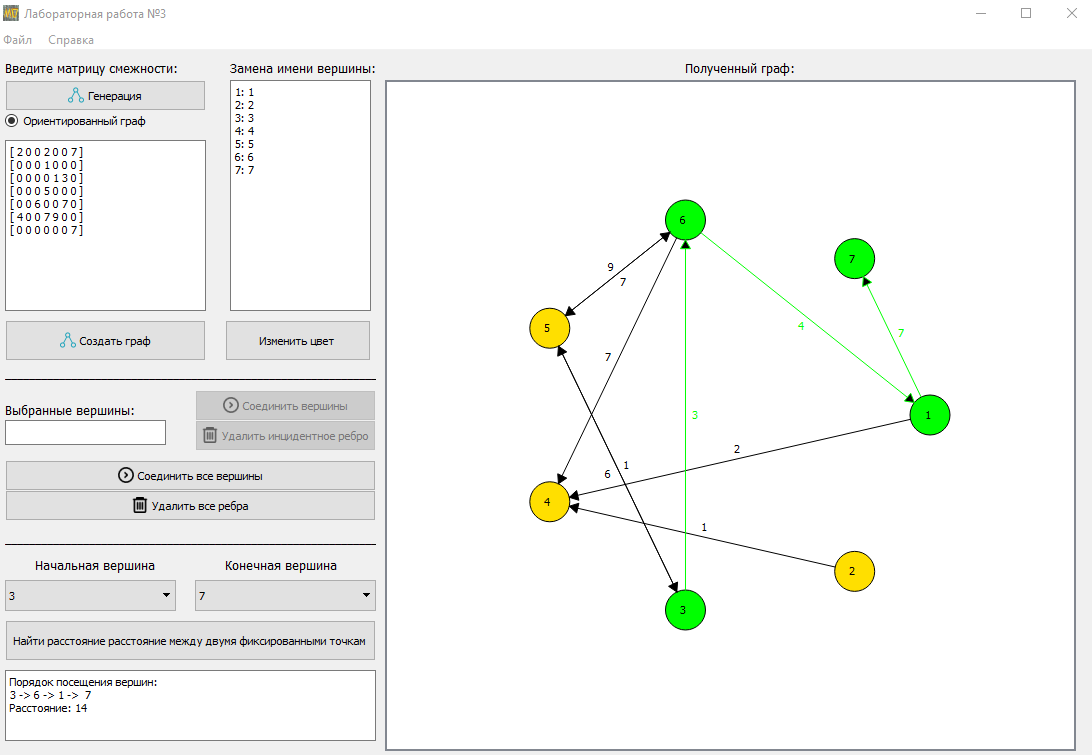


Рисунок 1. Интерфейс программы.

# **Вывод**

В ходе выполнения данной лабораторной работы были изучены основные понятия и приёмы применения алгоритма нахождения кратчайшего расстояния между любыми 2 вершинами ориентированного взвешенного графа

# **ПРИЛОЖЕНИЕ А.**

# **ИХОДНЫЙ КОД STRINGTOGRAPH.H**

#include <vector>

#include <iostream>

#include <QString>

#include <fstream>

using namespace std;

vector<vector<int>>readmatrix(QString);

QString **pathfinder**(int\*\* , int , int\*\* , int\*\* );

void **convertMatrix**(QString, int \*\*, int );

# **ПРИЛОЖЕНИЕ Б.**

# **ИХОДНЫЙ MAINWINDOW.H**

#pragma once

#ifndef MAINWINDOW\_H

#define MAINWINDOW\_H

#include "graphwidget.h"

namespace Ui {

class MainWindow;

}

class MainWindow : public QMainWindow

{

Q\_OBJECT

public:

explicit MainWindow(QWidget \*parent = 0);

~***MainWindow***();

private slots:

void **on\_pushButton\_clicked**();

void **on\_textEdit\_textChanged**();

void **on\_pushButton\_2\_clicked**();

vector<vector<MyNode>> **textEditToVector**();

void **on\_pushButton\_3\_clicked**();

void **updateMatrix**(const vector<vector<MyNode>>& );

void **on\_pushButton\_4\_clicked**();

void **on\_pushButton\_5\_clicked**();

void **on\_pushButton\_6\_clicked**();

void **on\_action\_triggered**();

void **on\_action\_2\_triggered**();

void **on\_action\_4\_triggered**();

void **on\_action\_3\_triggered**();

void **on\_pushButton\_7\_clicked**();

void **on\_radioButton\_clicked**();

void **on\_pushButton\_8\_clicked**();

void **on\_pushButton\_9\_clicked**();

private:

Ui::MainWindow \*ui;

};

#endif // MAINWINDOW\_H

# **ПРИЛОЖЕНИЕ В.**

# **ИХОДНЫЙ КОД GRAPHWIDGET.H**

#ifndef GRAPHWIDGET\_H

#define GRAPHWIDGET\_H

#include "edge.h"

#include "node.h"

#include "biconnected.h"

#include <QGraphicsView>

#include <QTextEdit>

#include <algorithm>

#include <QPushButton>

#include <QMessageBox>

#include <QFile>

#include <QFileDialog>

#include <QTextStream>

#include <QMainWindow>

#include <vector>

#include <map>

#include <QGraphicsScene>

#include <QGraphicsSceneMouseEvent>

#include <QPainter>

#include <QStyleOption>

#include <math.h>

#include <iostream>

#include <QPointF>

#include "helpbrowser.h"

using namespace std;

class Node;

struct MyNode

{

QString name;

int value;

QPointF pos;

QColor vv;

QColor vv2;

};

class GraphWidget : public QGraphicsView

{

Q\_OBJECT

public:

GraphWidget(vector<vector<MyNode>>, QTextEdit\* , int, QWidget \*parent = 0);

void **itemMoved**();

protected:

void ***timerEvent***(QTimerEvent \*event) override;

void **scaleView**(qreal scaleFactor);

private:

int timerId;

};

#endif // GRAPHWIDGET\_H

# **ПРИЛОЖЕНИЕ Г.**

# **ИХОДНЫЙ КОД EDGE.H**

#pragma once

#ifndef EDGE\_H

#define EDGE\_H

#include <QGraphicsItem>

#include <qmath.h>

#include <QPainter>

class Node;

class Edge : public QGraphicsItem

{

public:

Edge(Node \*sourceNode, Node \*destNode, int direction, QColor vv, int );

Node \***sourceNode**() const;

Node \***destNode**() const;

int direction;

void **adjust**();

enum { Type = UserType + 2 };

int ***type***() const override { return Type; }

protected:

QRectF ***boundingRect***() const override;

void ***paint***(QPainter \*painter, const QStyleOptionGraphicsItem \*option, QWidget \*widget) override;

private:

Node \*source, \*dest;

QPointF sourcePoint;

QPointF destPoint;

qreal arrowSize;

QColor color;

int value;

};

#endif // EDGE\_H

# **ПРИЛОЖЕНИЕ Д.**

# **ИХОДНЫЙ КОД NODE.H**

#pragma once

#ifndef NODE\_H

#define NODE\_H

#include <QGraphicsItem>

#include <QList>

#include <QTextEdit>

class Edge;

class GraphWidget;

class QGraphicsSceneMouseEvent;

class Node : public QGraphicsItem

{

public:

Node(QString, QTextEdit\* , QPointF, QColor, GraphWidget \*graphWidget);

void **addEdge**(Edge \*edge);

QList<Edge \*> **edges**() const;

enum { Type = UserType + 1 };

int ***type***() const override { return Type; }

QRectF ***boundingRect***() const override;

QPainterPath ***shape***() const override;

void ***paint***(QPainter \*painter, const QStyleOptionGraphicsItem \*option, QWidget \*widget) override;

QString maintext;

QTextEdit\* textEdit;

QColor color;

protected:

QVariant ***itemChange***(GraphicsItemChange change, const QVariant &value) override;

void ***mousePressEvent***(QGraphicsSceneMouseEvent \*event) override;

void ***mouseReleaseEvent***(QGraphicsSceneMouseEvent \*event) override;

void ***mouseDoubleClickEvent***(QGraphicsSceneMouseEvent \*event) Q\_DECL\_OVERRIDE;

void ***contextMenuEvent***(QGraphicsSceneContextMenuEvent \* event);

private:

QList<Edge \*> edgeList;

GraphWidget \*graph;

QPointF position;

};

#endif // NODE\_H

# **ПРИЛОЖЕНИЕ Е.**

# **ИХОДНЫЙ КОД HELPBROWSER.H**

#ifndef HELPBROWSER\_H

#define HELPBROWSER\_H

#include <QtWidgets>

//класс для отображения текстового представления

//справки

//или окна настроек

class HelpBrowser : public QDialog{

Q\_OBJECT

public:

HelpBrowser(const QString& strPath,const QString& strFileName, QWidget\* pwgt = 0 );

//конструктор для окна справки

static bool **outDialog**(const QString& text, QString title);

//функция для вызова текстового представления

private:

HelpBrowser(const QString& text\_view);

//конструктор для окна текстового представления

int new\_diam; //параметры для окна настройки

int arr\_size;

bool oriented;

QGraphicsScene\* scene\_for\_set;

QString name;

void **readSettings**();

void **writeSettings**();

private slots:

void **set\_new\_diam**(int a){new\_diam = a;} //установка настроек

void **set\_oriented**(bool b){oriented = b;}

void **set\_new\_arr**(int a){arr\_size = a;}

void **ok\_changes**();

void **appl\_changes**();

// void appl\_changes(); //применить настройки

protected:

void ***closeEvent***(QCloseEvent \* event);

};

#endif // HELPBROWSER\_H