**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 кратчайшее расстояние 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:

void MainWindow::**on\_pushButton\_3\_clicked**()

{

int qq=1;

int NN = sizeofarray;

int\* array = new int [NN];

QString l1,l2;

l1=ui->plainTextEdit\_2->toPlainText();

l2=ui->plainTextEdit\_3->toPlainText();

QString l=ui->plainTextEdit->toPlainText();

if (l.size()>10)

{

int N=0;

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

{

if (l[i]=='[')

++N;

}

int \*\*F = new int\* [N];

for (int i = 0; i < N; ++i)

F[i] = new int [N];

int \*\*next = new int\* [N];

for (int i = 0; i < N; ++i)

next[i] = new int [N];

int \*\*d = new int\* [N];

for (int i = 0; i < N; ++i)

d[i] = new int [N];

convertMatrix(l,F,N);

QString path = "|";

QString finded = pathfinder(F, N, next, d);

ui->plainTextEdit\_4->setPlainText(finded);

if ((l1.size()>0)&&(l2.size()>0))

{

int u,v;

bool ok1;

int n=l1.toInt(&ok1,10) -1 ;

if (ok1)

u=n;

n=l2.toInt(&ok1,10) - 1;

if (ok1)

v=n;

int c;

if (u>=N or v>=N)

{

ui->plainTextEdit\_5->setPlainText("No nodes found");

QPixmap image("main.png");

ui->label->setPixmap(image);

ui->scrollArea->setWidget(ui->label);

return;

}

if (d[u][v] == 0)

{

ui->plainTextEdit\_5->setPlainText("No path found");

QPixmap image("main.png");

ui->label->setPixmap(image);

ui->scrollArea->setWidget(ui->label);

return;

}

else

{

c = u;

while (c != v)

{

if (c==u)

{

path=QString::number(c+1);

array[qq] = c;

qq++;

}

else

{

path+=QString::number(c+1);

array[qq] = c;

qq++;

}

path+="-> ";

c = next[c][v];

}

path+=QString::number(v+1);

array[qq] = v;

qq++;

path+='=';

path+=QString::number(d[u][v]);

ui->plainTextEdit\_5->setPlainText(path);

}

}

array[0] = qq;

newrender(F, N, array);

for (int i = 0; i < N; ++i)

delete []d[i];

for (int i = 0; i < N; ++i)

delete []next[i];

for (int i = 0; i < N; ++i)

delete []F[i];

}

else

ui->plainTextEdit\_2->setPlainText("error");

}

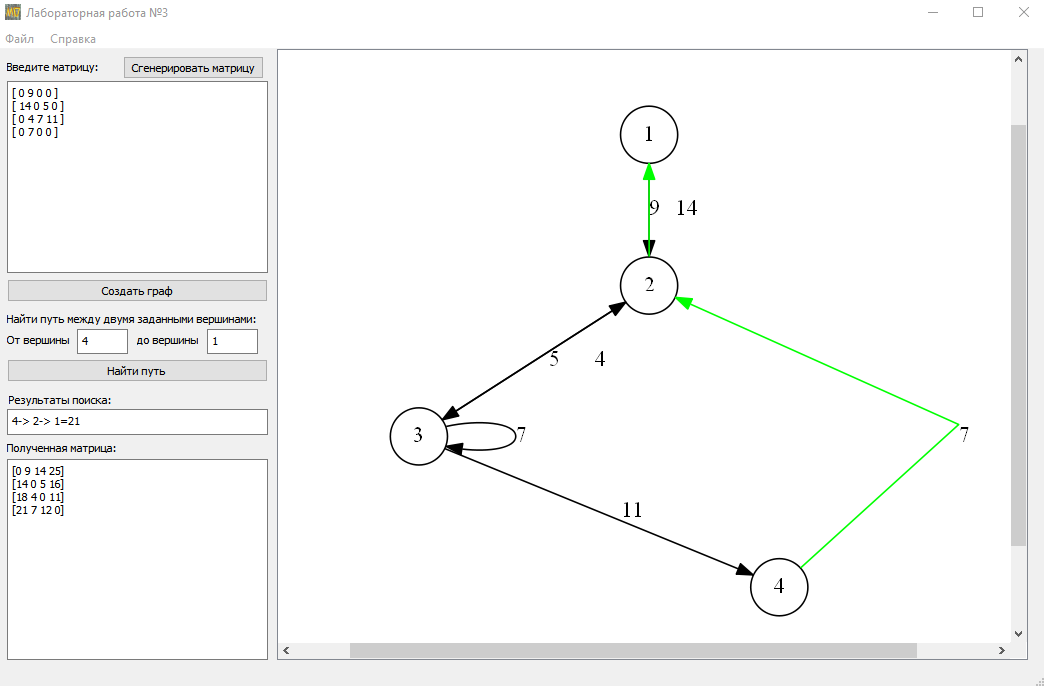


Рисунок 1. Интерфейс программы.

# **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 );

void **generateDotFile**(int\*\* , int, int\*, string);

# **ПРИЛОЖЕНИЕ Б.**

# **ИХОДНЫЙ КОД STRINGTOGRAPH.CPP**

#include "stringtograph.h"

vector<vector<int>>**readmatrix**(QString x)

{

QString stringMatrix = x;

int size=0;

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

{

if (stringMatrix[i]=='[')

++size;

}

int j=0; int k=0;

vector<vector<int>> matrix(size, vector<int>(size, 0));

QString currentValue;

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

{

if ((stringMatrix[i]!='[') &&

(stringMatrix[i]!=']') &&

(stringMatrix[i]!=' ') &&

(stringMatrix[i]!='\n'))

{

if ((stringMatrix[i-1]=='[')||(stringMatrix[i-1]==' '))

currentValue=stringMatrix[i];

else

currentValue+=stringMatrix[i];

if ((stringMatrix[i+1]==']')||(stringMatrix[i+1]==' '))

{

bool checkCorretConvert;

int curValue=currentValue.toInt(&checkCorretConvert,10);

if (checkCorretConvert)

matrix[k][j]=curValue;

if (j==size-1)

{

j=0;

++k;

}

else

++j;

}

}

}

return matrix;

}

QString **pathfinder**(int \*\*F, int N, int \*\*next, int \*\*d)

{

QString x;

for (int i=0;i<N;++i)

{

for (int j=0;j<N;++j)

{

if (F[i][j]!=0)

next[i][j]=j;

else

next[i][j]=0;

}

}

for (int i=0;i<N;++i)

{

for (int j=0;j<N;++j)

{

d[i][j]=F[i][j];

if (F[i][j]==0)

d[i][j]=1000;

}

}

for (int i=0;i<N;++i)

{

for (int u=0;u<N;++u)

{

for (int v=0;v<N;++v)

{

if (d[u][i] + d[i][v] < d[u][v])

{

d[u][v] = d[u][i] + d[i][v];

next[u][v] = next[u][i];

}

}

}

}

for (int i=0;i<N;++i)

{

for (int j=0;j<N;++j)

{

if ((i==0)&&(j==0))

x='[';

else

if (j==0)

x+='[';

if (d[i][j]==1000)

d[i][j]=0;

if (i==j)

d[i][j]=0;

x+=QString::number(d[i][j]);

if (j==N-1)

{

x+=']';

x+='\n';

}

else

x+=' ';

}

}

return x;

}

void **convertMatrix**(QString l, int \*\*L, int N)

{

QString x;

int j=0; int k=0;

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

{

if ((l[i]!='[')&&(l[i]!=']')&&(l[i]!=' ')&&(l[i]!='\n'))

{

if ((l[i-1]=='[')||(l[i-1]==' '))

{

x=l[i];

}

else

{

x+=l[i];

}

if ((l[i+1]==']')||(l[i+1]==' '))

{

bool ok;

int n=x.toInt(&ok,10);

if (ok)

L[k][j]=n;

if (j==N-1)

{

j=0;

++k;

}

else

++j;

}

}

}

}

void **generateDotFile**(int \*\*matrix, int N, int\* array, string headerDot)

{

string readout;

ofstream f("dot1.dot");

f << headerDot;

for (int i=0;i<N;++i)

{

for (int j=0;j<N;++j)

{

if (matrix[i][j]>0)

{

QString pathQString = QString::number(i+1) + " -> " + QString::number(j+1)

+ " [label=\"" + QString::number(matrix[i][j]) + "\"];\n";

string pathString = pathQString.toLocal8Bit().constData();

f << pathString;

}

}

}

f << "}";

f.close();

for (int i=1; i< array[0] -1 ; i++)

{

if (array[i] >= 0)

{

int ii = array[i] + 1;

int jj = array[i+1] + 1;

ofstream outFile("dot2.dot");

ifstream readFile("dot1.dot");

QString searchx = QString::number(ii) + " -> " + QString::number(jj);

string search = searchx.toLocal8Bit().constData();

while(getline(readFile,readout))

{

if(readout.find(search) != std::string::npos)

{

QString x = QString::number(ii) + " -> " + QString::number(jj) +

" [label=\"" + QString::number(matrix[ii-1][jj-1]) +"\" color = \"green\" ];\n";

string xx = x.toLocal8Bit().constData();

outFile << xx;

}

else

outFile << readout << "\n";

}

outFile.close();

readFile.close();

ofstream outFile1("dot1.dot");

ifstream readFile1("dot2.dot");

while(getline(readFile1, readout))

{

outFile1 << readout << "\n";

}

outFile1.close();

readFile1.close();

matrix[ii-1][jj-1] = -1;

}

}

}

# **ПРИЛОЖЕНИЕ В.**

# **ИХОДНЫЙ MAINWINDOW.H**

#ifndef MAINWINDOW\_H

#define MAINWINDOW\_H

#include <QMainWindow>

namespace Ui {

class MainWindow;

}

class MainWindow : public QMainWindow

{

Q\_OBJECT

public:

explicit MainWindow(QWidget \*parent = 0);

~***MainWindow***();

private slots:

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

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

void **newrender**(int \*\*F, int N, int\* array);

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

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

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

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

private:

Ui::MainWindow \*ui;

};

#endif // MAINWINDOW\_H

# **ПРИЛОЖЕНИЕ Г.**

# **ИХОДНЫЙ КОД MAINWINDOW.CPP**

#include "mainwindow.h"

#include "ui\_mainwindow.h"

#include <QMessageBox>

#include <QFileDialog>

#include <QPlainTextEdit>

#include "stringtograph.h"

MainWindow::**MainWindow**(QWidget \*parent) :

QMainWindow(parent),

ui(new Ui::MainWindow)

{

ui->setupUi(this);

}

MainWindow::~***MainWindow***()

{

delete ui;

}

int sizeofarray = 0;

string headerDot = "digraph { graph [pad=\"1.2\", nodesep=\"0.6\", ranksep=\"0.6\"]; "

"splines=\"false\"; "

"node[shape = circle]; "

"edge[style=rounded]; "

"ratio=\"fill\"; "

"margin=0;\n";

void MainWindow::**on\_createGraph\_clicked**()

{

ofstream f("dot.dot");

f << headerDot;

vector<vector<int>> b = readmatrix(ui->plainTextEdit->toPlainText());

sizeofarray = b[0].size();

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

for (int j=0; j< b.size(); j++)

{

if (b[i][j] != 0)

{

QString pathQString = QString::number(i+1) + " -> " + QString::number(j+1) + " [label=\"" + QString::number(b[i][j]) + "\"];\n";

string pathString = pathQString.toLocal8Bit().constData();

f << pathString;

}

}

f << "}";

f.close();

system("CD /D \"D:\Documents\Qt Projects\build-graphviz\_"

"beta-Desktop\_Qt\_5\_10\_0\_MinGW\_32bit-Debug\" & "

"dot -Gsize=3,3\! -Gdpi=300 -Tpng dot.dot -o main.png");

QPixmap image("main.png");

ui->label->setPixmap(image);

ui->label->repaint();

ui->scrollArea->setWidget(ui->label);

window()->setFixedSize(1050, 660);

}

void MainWindow::**on\_pushButton\_2\_clicked**()

{

QString stringMatrix = "";

int size = rand()% 5 + 3;

vector<vector<int>> matrix(size, vector<int>(size, 0));

for (int i=0; i< size; i++)

{

stringMatrix += "[ ";

for (int j=0; j<size; j++)

{

int randomize = rand()% 3;

if (!randomize)

matrix[i][j] = rand() % 20;

stringMatrix = stringMatrix + QString::number(matrix[i][j]) + " ";

}

stringMatrix += "]\n";

}

ui->plainTextEdit->setPlainText(stringMatrix);

}

void MainWindow::**newrender**(int \*\*F, int N, int\* array)

{

generateDotFile(F, N, array, headerDot);

system("CD /D \"D:\Documents\Qt Projects\build-graphviz\_beta-Desktop\_Qt\_5\_10\_0\_MinGW\_32bit-Debug\" & dot -Gsize=3,3\! -Gdpi=300 -Tpng dot2.dot -o rendered.png");

QPixmap image("rendered.png");

ui->label->setPixmap(image);

ui->scrollArea->setWidget(ui->label);

}

void MainWindow::**on\_pushButton\_3\_clicked**()

{

int qq=1;

int NN = sizeofarray;

int\* array = new int [NN];

QString l1,l2;

l1=ui->plainTextEdit\_2->toPlainText();

l2=ui->plainTextEdit\_3->toPlainText();

QString l=ui->plainTextEdit->toPlainText();

if (l.size()>10)

{

int N=0;

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

{

if (l[i]=='[')

++N;

}

int \*\*F = new int\* [N];

for (int i = 0; i < N; ++i)

F[i] = new int [N];

int \*\*next = new int\* [N];

for (int i = 0; i < N; ++i)

next[i] = new int [N];

int \*\*d = new int\* [N];

for (int i = 0; i < N; ++i)

d[i] = new int [N];

convertMatrix(l,F,N);

QString path = "|";

QString finded = pathfinder(F, N, next, d);

ui->plainTextEdit\_4->setPlainText(finded);

if ((l1.size()>0)&&(l2.size()>0))

{

int u,v;

bool ok1;

int n=l1.toInt(&ok1,10) -1 ;

if (ok1)

u=n;

n=l2.toInt(&ok1,10) - 1;

if (ok1)

v=n;

int c;

if (u>=N or v>=N)

{

ui->plainTextEdit\_5->setPlainText("No nodes found");

QPixmap image("main.png");

ui->label->setPixmap(image);

ui->scrollArea->setWidget(ui->label);

return;

}

if (d[u][v] == 0)

{

ui->plainTextEdit\_5->setPlainText("No path found");

QPixmap image("main.png");

ui->label->setPixmap(image);

ui->scrollArea->setWidget(ui->label);

return;

}

else

{

c = u;

while (c != v)

{

if (c==u)

{

path=QString::number(c+1);

array[qq] = c;

qq++;

}

else

{

path+=QString::number(c+1);

array[qq] = c;

qq++;

}

path+="-> ";

c = next[c][v];

}

path+=QString::number(v+1);

array[qq] = v;

qq++;

path+='=';

path+=QString::number(d[u][v]);

ui->plainTextEdit\_5->setPlainText(path);

}

}

array[0] = qq;

newrender(F, N, array);

for (int i = 0; i < N; ++i)

delete []d[i];

for (int i = 0; i < N; ++i)

delete []next[i];

for (int i = 0; i < N; ++i)

delete []F[i];

}

else

ui->plainTextEdit\_2->setPlainText("error");

}

void MainWindow::**on\_openfile\_triggered**()

{

QString FileName = QFileDialog::getOpenFileName(this, "Открыть файл...", QDir::homePath(), "Text files (\*.txt)");

if (FileName.isEmpty())

return;

QFile File(FileName);

if(File.*open*(QIODevice::ReadOnly | QIODevice::Text))

{

ui->plainTextEdit->setPlainText(File.readAll());

File.*close*();

}

else

{

QMessageBox::warning(this, "Error", QString("Не смог открыть файл %1").arg(FileName),QMessageBox::Ok);

}

}

void MainWindow::**on\_action\_triggered**()

{

QMessageBox::about(this, tr("О программе"), tr("Нахождение кратчайшего пути между\nдвумя вершинами."));

return;

}

void MainWindow::**on\_action\_2\_triggered**()

{

QMessageBox::about(this, tr("Об авторе"), tr("Лабораторная работа №3\n\nВыполнил: Ковынев М.В.\nГруппа: 6304"));

return;

}

# **ПРИЛОЖЕНИЕ Д.**

# **ИХОДНЫЙ КОД MAIN.CPP**

#include "mainwindow.h"

#include <QApplication>

#include <time.h>

int main(int argc, char \*argv[])

{

QApplication a(argc, argv);

MainWindow w;

w.show();

w.setFixedSize(280, 300);

w.setWindowIcon(QIcon("2.png"));

w.setWindowTitle("Лабораторная работа №3");

srand(time(NULL));

return a.exec();