**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра МО ЭВМ**

отчет

**по лабораторной работе №3**

**по дисциплине «Построение и анализ алгоритмов»**

Тема: Алгоритм Дейкстры

|  |  |  |
| --- | --- | --- |
| Студент гр. 6304 |  | Пискунов Я.А. |
| Преподаватель |  | Балтрашевич В.Э. |

Санкт-Петербург

2018

**Цель работы.**

Изучить и реализовать алгоритм Дейкстры на языке C++

**Основные теоретические положения.**

*Алгоритм Дейкстры* – алгоритм нахождения минимального расстояния от одной вершины графа до остальных.

За основу возьмем алгоритм на языке Paskal, представленный в книге Липского «Комбинаторика для программистов»:

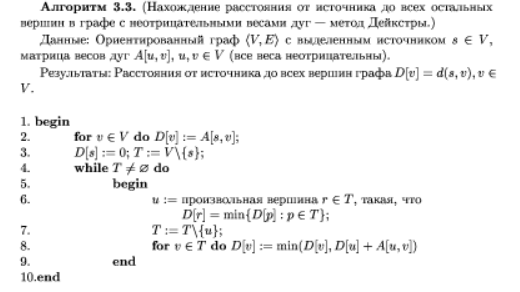


Рисунок 1- псевдокод алгоритма

Так как в реализации данного алгоритма необходимо работать со взвешенным графом, то переработаем структуру данных, в которой содержится граф.

**Ход работы**

Граф хранится с помощью двух структур. В первой структуре хранятся элементы графа, а во второй – его сыновья. Каждая структура содержит указатель на своего представителя в графической сцене. Ребра содержат не только указатель на второй элемент, но и его имя. Это связано с тем, что при вводе из файла может быть дана ссылка на вершину, которой ещё нет в основном списке. В таком случае записывается только имя, и зависимости разрешаются позднее. В отличие от предыдущей лабораторной, теперь ребра поддерживают веса. Кроме того, пометки на ребрах теперь могут быть разных порядков. Структуры представлены в приложении А.

Для удобства работы с графом создадим класс, в который поместим все необходимые переменные и структуры, а также методы работы с ним. Заголовочный файл с данными по этому классу представлен в приложении Б. Рассматривать подробно детали реализации графа в данном отчете не будем, так как они не являются главной задачей данной работы. Комментарии к реализации указанных методов есть непосредственно в коде.

Помимо всего прочего, класс включает в себя макросы для сохранения и восстановления итераторов:

#define SAVEITS List\* t\_lpos = KeepItL(); Elem\* t\_pos = KeepItE(); int t\_linpos = linpos;

#define RESTOREITS RestoreItL(t\_lpos); RestoreItE(t\_pos); linpos = t\_linpos;

Они нужны для того, чтобы запуск методов класса не изменял итераторы, запущенные на уровень выше.

Вид интерфейса представлен на рис. 2. Аспекты работы с интерфейсом изложены внутри самой программы и вызываются по нажатию кнопки F1 или выбора пункта Help в меню About. Они изложены в виде набора html-страниц, по которым можно перемещаться.

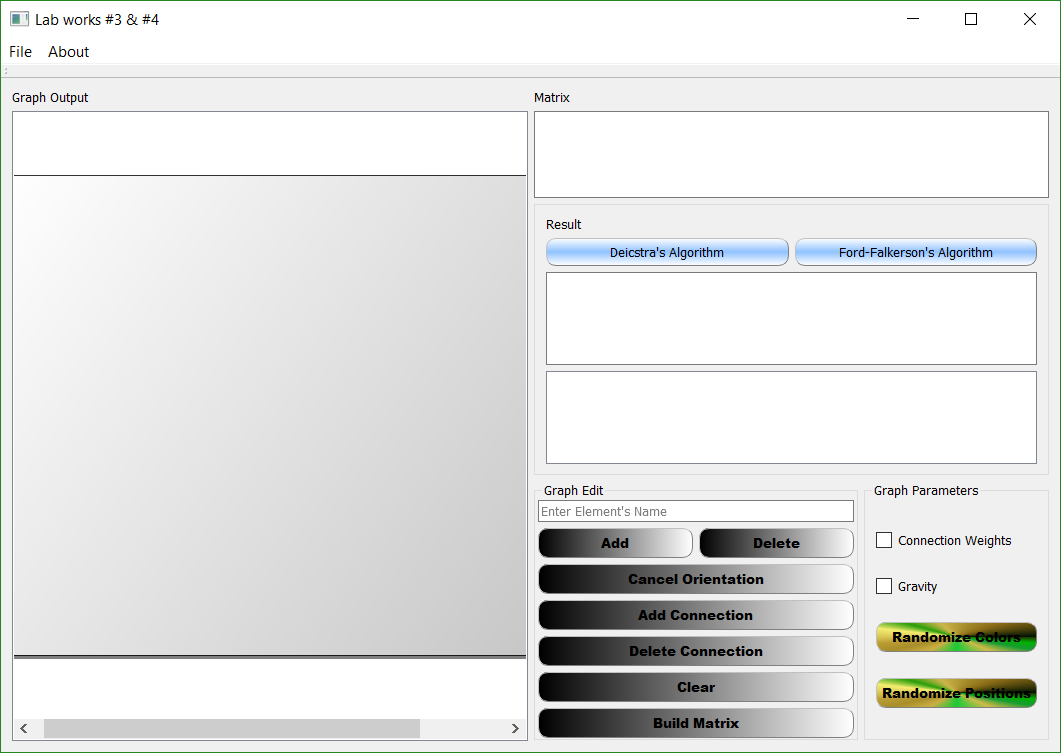


Рисунок 2 – интерфейс программы

Программа может как считывать готовый граф из файла определенной структуры, так и работать с созданным с нуля с помощью QGraphicsScene графом. Нюансы работы с ним также описаны в меню Help. Программа поддерживает возможность редактирования графа, добавление и удаление вершин и ребер из него. Кроме этого, в ней реализованы два алгоритма. В данной работе внимание уделим алгоритму Дейкстры. Прежде чем запустить алгоритм, необходимо выбрать исходную вершину(исток) и конечную вершину(сток), относительно которых будет работать алгортим. Запуск алгоритма осуществляется по нажатию соответствующей кнопки в интерфейсе. Код, который при этом выполняется представлен в приложении В.

Так как при выполнении алгоритма необходимо взаимодействовать с графом, то целесообразно добавить алгоритм как один из публичных методов в класс графа. Сам алгоритм состоит из двух частей: инициализация с проверкой возможности выполнения алгоритма и непосредственно сам алгоритм. Эти части соответственно сделаны в виде отдельных методов. Они представлены в приложении Г.

Также необходимо как-то выводить результат на экран. Формат вывода результата следующий. Формируется таблица из двух строк. В первую записываются имена вершин, во вторую – расстояния от выбранной вершины до них. В случае невозможности попасть из выбранной вершины в некоторую, то в качестве расстояния до нее выводится значение INT\_MAX.

Теперь проведем несколько тестов программы. Для этого возьмем пример из все той же книги Липского «Комбинаторика для программистов», также один простой пример. Результаты работы программы представлены на рис. 3 и рис. 4 (Вершина, относительно которой считаются пути обозначена красным).

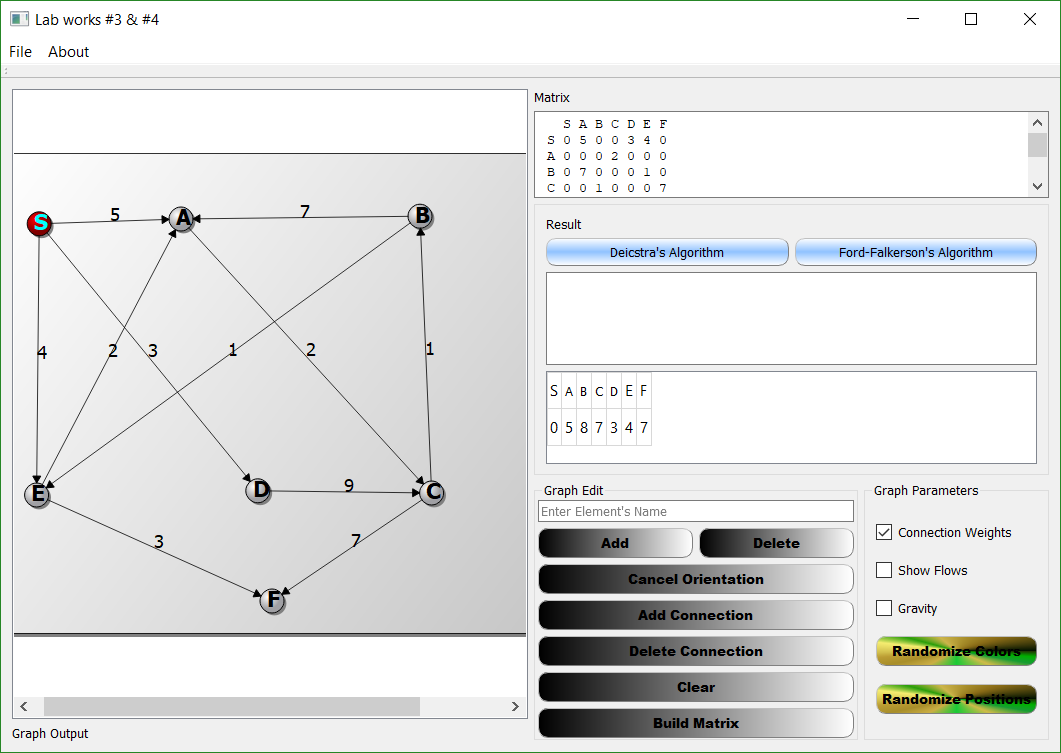


Рисунок 3 – пример работы программы

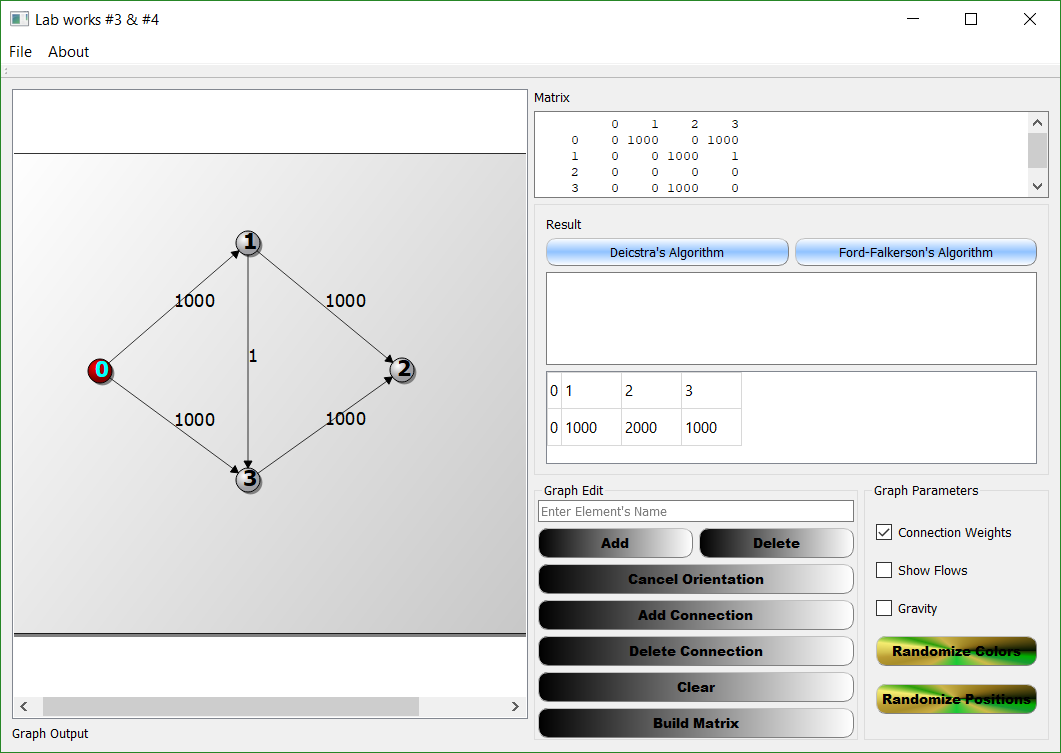


Рисунок 4 – пример работы программы

Как мы видим, результаты работы программы соответствуют представленным в книге и здравому смыслу, что говорит о правильности алгоритма.

**Выводы.**

При выполнении данной работы изучен алгоритм Дейкстры и его реализация на языке C++. Изучены вопросы повышения эффективности работы алгоритма.

Исследованы возможности использования ресурсов в Qt для работы с HTML-документами и картинками в программе.

Приложение А

Структура графа

typedef struct Elem{

Elem() = default;

char name[Numb];

Elem\* next=nullptr;

List\* childs=nullptr;

Node\* node;

bool searched = false;

}Elem;

typedef struct List{

List() = default;

char name[Numb];

int mark = 0;

int weight = 1;

int flow = 0;

Elem\* node = nullptr;

List\* next = nullptr;

Edge\* edge;

}List;

Приложение Б

Код graph.H

#ifndef GRAPH\_H

#define GRAPH\_H

#define SAVEITS List\* t\_lpos = KeepItL(); Elem\* t\_pos = KeepItE(); int t\_linpos = linpos;

#define RESTOREITS RestoreItL(t\_lpos); RestoreItE(t\_pos); linpos = t\_linpos;

#define Numb 80

#define Wid 800

#define Hei 600

#include <iostream>

#include <cstring>

#include <climits>

#include <QTextStream>

#include <QString>

#include <QFile>

#include <QStack>

#include <QMessageBox>

#include <QInputDialog>

#include <QTime>

#include "graphwidget.h"

using namespace std;

typedef struct List List;

typedef struct Elem{

Elem() = default;

char name[Numb];

Elem\* next=nullptr;

List\* childs=nullptr;

Node\* node;

bool searched = false;

}Elem;

typedef struct List{

List() = default;

char name[Numb];

int mark = 0;

int weight = 1;

int flow = 0;

Elem\* node = nullptr;

List\* next = nullptr;

Edge\* edge;

}List;

typedef struct Path{

QList<List\*> way;

QList<Elem\*> way\_el;

int min\_flow = INT\_MAX / 2;

}Path;

class Graph

{

public:

Graph();

~Graph();

void **Clear**();

Elem\* **it**();

List\* **it**(Elem\* el, bool marked = 1);

List\* **itin**(Elem\* el);

void **ResetIts**();

void **ReadFile**(QString fileName);

void **SaveFile**(QString fileName, bool pos = 0);

char\* **GetMinStupidName**();

char\* **GetLastStupidName**();

void **AddElem**(char\* name, double X = -100, double Y = -100);

void **AddEdge**(Elem\* el1, Elem\* el2, int weight = 1);

void **AddEdge**(Elem\* el1, char\* name, int weight = 1);

bool **Solve**();

void **RemoveElem**(char\* name);

void **RemoveEdge**(Elem\* el1, Elem\* el2);

void **RemoveEdges**(Elem\* el);

void **RenameElem**(char\* oldname, char\* newname);

void **Desorientate**();

Elem\* **FindElem**(char\* name);

int **CountChildren**(Elem\* el, bool marked = 1);

int **CountElems**();

int **Is\_Egde**(Elem\* el1, Elem\* el2, bool noabs = 0);

List\* **GetEdge**(Elem\* el1, Elem\* el2);

int **Max\_Width**();

void **Inc\_Matr**(QTextStream& os);

void **Mark**(List\* ls, int mark = 0);

void **ClearMarks**();

Elem\* **FindPath**(Elem\* el);

void **ResetFindPath**();

void **SetMinPath**();

bool **DeicstraInit**(Elem \*v0i = nullptr);

void **DeicstraAlgorithm**();

QQueue<List\*> **MakeQueue**(Elem\*\* previous);

Elem\* **FindMinimal**(QQueue<List \*> \*H);

void **FordInit**();

bool **FordAlgorithm**();

void **WeightsOn**(bool state);

void **ChangeWeight**(Elem\* el1, Elem\* el2, int weight);

double **AverageWeight**();

void **ClearWeights**();

Elem\* **operator**[](int i);

int **number**(Elem\* el);

GraphWidget\* widget;

Elem\* marked = nullptr;

bool itermarks = false;

bool weights = false;

Elem\* source = nullptr;

Elem\* sink = nullptr;

Path path;

int max\_iter = 0;

Elem\* v0 = nullptr;

int\* arr = nullptr;

int ib = 0;

int vm = 10;

int changes = 0;

bool go = 1;

private:

int it\_num = 0;

int stupidnames = 0;

Elem\* gr;

Elem\* pos;

List\* lpos;

int linpos;

void **Clear**(List\* ls, Elem \*el);

void **Clear**(Elem\* gr);

Elem\* **KeepItE**();

List\* **KeepItL**();

void **RestoreItE**(Elem\* t\_pos);

void **RestoreItL**(List\* t\_lpos);

};

#endif // GRAPH\_H

Приложение В

Код запуска алгоритма

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

{

bool ok = !(gr1->DeicstraInit(gr1->v0));

if (ok)

{

gr1->DeicstraAlgorithm();

DeicstraResult(gr1->arr);

}

else

{

QMessageBox box;

box.setText("No Element Chosen");

box.*exec*();

}

}

Приложение Г

Код Методов алгоритма

void Graph::**DeicstraAlgorithm**()

{

SAVEITS;

int ammount = this->CountElems();

arr[number(this->FindElem(v0->name))] = 0;

Elem\*\* previous = new Elem\*[ammount];

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

previous[i] = nullptr;

QQueue<List\*> H = MakeQueue(previous);

while (!H.isEmpty())

{

Elem\* v = FindMinimal(&H);

List\* u\_temp = it(v);

while (u\_temp)

{

Elem\* u = this->FindElem(u\_temp->name);

if (arr[number(u)] > (arr[number(v)] + (GetEdge(v, u))->weight))

{

arr[number(u)] = arr[number(v)] + (GetEdge(v, u))->weight;

previous[number(u)] = v;

//Update Priorities

H.enqueue(u\_temp);

}

u\_temp = it(v);

}

}

RESTOREITS;

}

bool Graph::**DeicstraInit**(Elem\* v0i)

{

if (v0i!=nullptr)

v0 = v0i;

else{

bool ok;

QString namev0 = QInputDialog::getText(widget, "Deicstra's Algorithm", "Please Enter Name of the First Element", QLineEdit::Normal, "", &ok);

QByteArray arr = namev0.toLocal8Bit();

char\* cnamev0 = arr.data();

v0 = FindElem(cnamev0);

}

if (v0 == nullptr){

QMessageBox msg;

msg.setText("Error 404: First Element not in Graph!");

msg.*exec*();

return 1;

}

if (arr!=nullptr){

delete[] arr;

arr = nullptr;

}

vm = CountElems();

if (vm == 0)

return 1;

arr = new int[vm];

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

arr[i] = INT\_MAX/2;

}

arr[number(v0)] = 0;

ib = 0;

ResetIts();

return 0;

} Приложение Д

Вывод результата на экран

void MainWindow::**DeicstraResult**(int\* data)

{

int n = gr1->CountElems();

int i;

ui->resTableWidget->clear();

ui->resTableWidget->setRowCount(2);

ui->resTableWidget->setColumnCount(n);

for (i=0; i<n; i++)

{

QTableWidgetItem \*item = new QTableWidgetItem;

item->setText(gr1->operator [](i)->name);

ui->resTableWidget->setItem(0, i, item);

QTableWidgetItem \*item2 = new QTableWidgetItem;

QString str = QString::number(data[i]);

item2->setText(str);

ui->resTableWidget->setItem(1, i, item2);

ui->resTableWidget->setColumnWidth(i, strlen(gr1->operator [](i)->name)\*15);

}

}