

МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МО ЭВМ

ОТЧЕТ
по учебной практике
ТЕМА: «Алгоритм Дейкстры поиска кратчайших путей в графе»

Студент гр. 8382	_____	Кобенко В.П
Студент гр. 8382	_____	Вербин К.М.
Студентка гр. 8382	_____	Ефимова М.А.
Руководитель	_____	Фирсов М.А.

Санкт-Петербург
2020

ЗАДАНИЕ НА УЧЕБНУЮ ПРАКТИКУ

Студент Кобенко В.П. группы 8382

Студент Вербин К.М. группы 8382

Студентка Ефимова М.А. группы 8382

Тема практики: алгоритм Дейкстры

Задание на практику:

Командная итеративная разработка визуализатора алгоритма(ов) на Java с графическим интерфейсом.

Алгоритм: Дейкстры

Сроки прохождения практики: 29.06.2020 – 12.07.2020

Дата сдачи отчета: 12.07.2020

Дата защиты отчета: 00.07.2020

Студент	_____	Кобенко В.П.
Студент	_____	Вербин К.М.
Студентка	_____	Ефимова М.А.
Руководитель	_____	Фирсов М.А.

АННОТАЦИЯ

Целью учебной практики является разработка приложения для визуализации алгоритма Дейкстры. Приложение создается на языке Java и должно обладать графическим интерфейсом. Пользователю должна быть предоставлена возможность отрисовки используемых структур данных (графа и соответствующей матрицы смежности), а также пошагового выполнения алгоритма с пояснениями. Приложение должно быть понятным и удобным для использования.

Задание выполняется командой из трех человек, за которыми закреплены определенные роли. Выполнение работы и составление отчета осуществляются поэтапно.

SUMMARY

The purpose of training practice is to create an application which would visualize the Dijkstra's algorithm. The application should be written in Java programming language and must implement a graphical user interface. The user must be provided with possibilities to view data structures in use (the graph and the respective adjacency matrix). The application must be transparent and handy.

The task is fulfilled by a team of three members, each of them assigned with certain obligations. Implementation of the task and report composition should be gradual.

СОДЕРЖАНИЕ

	Введение	5
1.	Требования к программе	6
1.1.	Требования к вводу исходных данных	6
1.2.	Требования к выводу результата	6
1.3.	Требования к интерфейсу и вводу графа через графический интерфейс	6
1.4	Требования к визуализации	7
2.	План разработки и распределение ролей в бригаде	9
2.1.	План разработки	9
2.2.	Распределение ролей в бригаде	9
3.	Особенности реализации	10
3.1.	Структура данных	10
3.2.	Основные методы	10
3.3	Версия 0. Прототип	10

ВВЕДЕНИЕ

Целью учебной практики является создание приложения, визуализирующего работу алгоритма Дейкстры, предназначенного для нахождения всех кратчайших путей взвешенного графа с неотрицательными весами из исходной вершины до каждой вершины графа, последовательно наращивая множество вершин, для которых известен кратчайший путь. Приложение должно быть написано на языке Java и снабжено понятным и удобным в использовании графическим интерфейсом. Пользователю должна быть предоставлена возможность ввести исходные данные в самой программе с использованием клавиатуры и мыши. Результат работы алгоритма также должен выводиться на экран. Должна быть предоставлена возможность моментального отображения результата.

Задание выполняется командой из трех человек, за каждым из которых закреплены определенные обязанности – реализация графического интерфейса, логики алгоритма, проведение тестирования и сборка проекта. Готовая программа должна корректно собираться из исходников в один исполняемый jar-архив. В ходе сборки должны выполняться модульные тесты и завершаться успехом. Также на момент завершения практики должен быть составлен подробный отчет, содержащий моделирование программы, описание алгоритмов и структур данных, план тестирования, исходный код и др.

1. ТРЕБОВАНИЯ К ПРОГРАММЕ

1.1. Требования к вводу исходных данных

Исходными данными для реализуемого приложения является неориентированный граф с положительными весами, в котором будет осуществляться поиск кратчайшего пути между вершинами и стартовая вершина. Граф задается списком ребер в формате $v_i v_j w_{ij}$, где v_i, v_j – смежные вершины, w_{ij} – вес (длина) ребра между ними. Пользователь сможет ввести данные как из файла, так и с помощью графического интерфейса. Будет реализовано редактирование графа в графическом интерфейсе.

1.2. Требования к выводу результата

Результат выполнения алгоритма должен выводиться на экран в виде таблицы и графически. При запуске алгоритма пользователь сможет просмотреть кратчайшие пути от стартовой вершины до любой вершины в заданном графе. В ходе работы приложения пользователь сможет пошагово просмотреть работу алгоритма. Работу алгоритма будет возможно просмотреть как в консоли, так и в графическом интерфейсе.

1.3. Требования к интерфейсу и вводу графа через графический интерфейс

Необходимо реализовать удобный и понятный пользователю графический интерфейс. Должна быть предоставлена возможность отрисовки заданного графа, выполнение алгоритма по требованию пользователя необходимо осуществлять моментально с выводом результата.

Пользователю будет дана возможность ввода графа с помощью мыши (клик левой мыши – указание положение вершин на плоскости графа, клик правой мыши – выделение стартовой и финальной вершины; зажим левой кнопки мыши между вершинами – соединение вершин ребром), ввод с помощью списка смежности из файла. Также будет реализована стартовая кнопка, запускающая работу алгоритма Дейкстры и кнопка очистки рабочей области.

Выведется окно, в котором будет область для рисования графа и следующие кнопки: перезапуск, запуск, ввод из файла и информация.

1.4. Требования к визуализации

Во время работы алгоритма в консоль в текстовом формате будут выведены шаги работы алгоритма. Также на обрисованном графе будут продемонстрированы шаги работы алгоритма.

В консоли будут выведены шаги работы алгоритма, содержащие просмотренные вершины, вершину, которую обрабатывает алгоритм на данном шаге, вершины которые смежны с просматриваемой вершиной и изменения значений меток для этих вершин. На каждом шаге будут выделены просматриваемая вершина и путь до нее. Около вершин будут написаны значения меток.

На каждом шаге будут раскрашены одним цветом просмотренные вершины, другим цветом – не просмотренные вершины, а третьим цветом – текущая вершина. Текущий путь от начальной вершины до просматриваемой будет выделен цветом просматриваемой вершины.

На рисунке 1 изображена UML – диаграмма проекта.

На рисунке 2 изображен эскиз интерфейса.

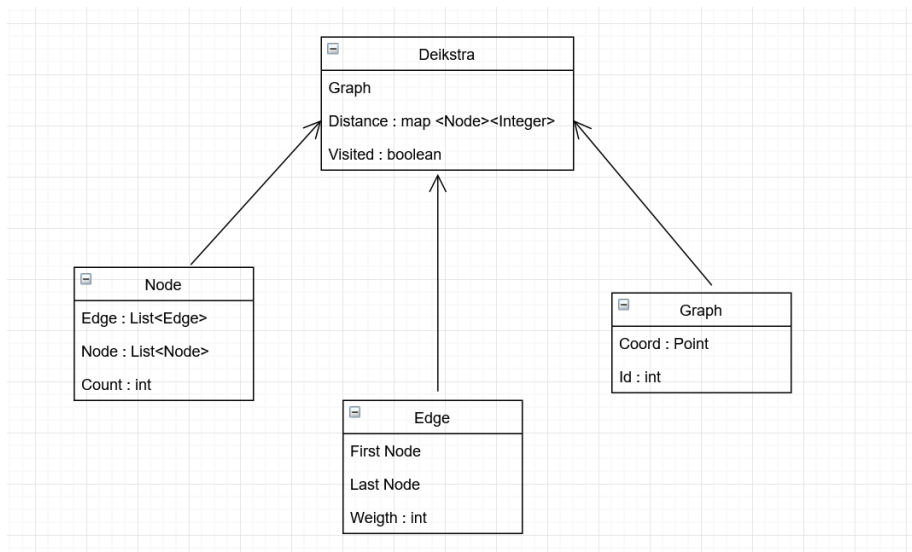


Рисунок 1 – UML – диаграмма.

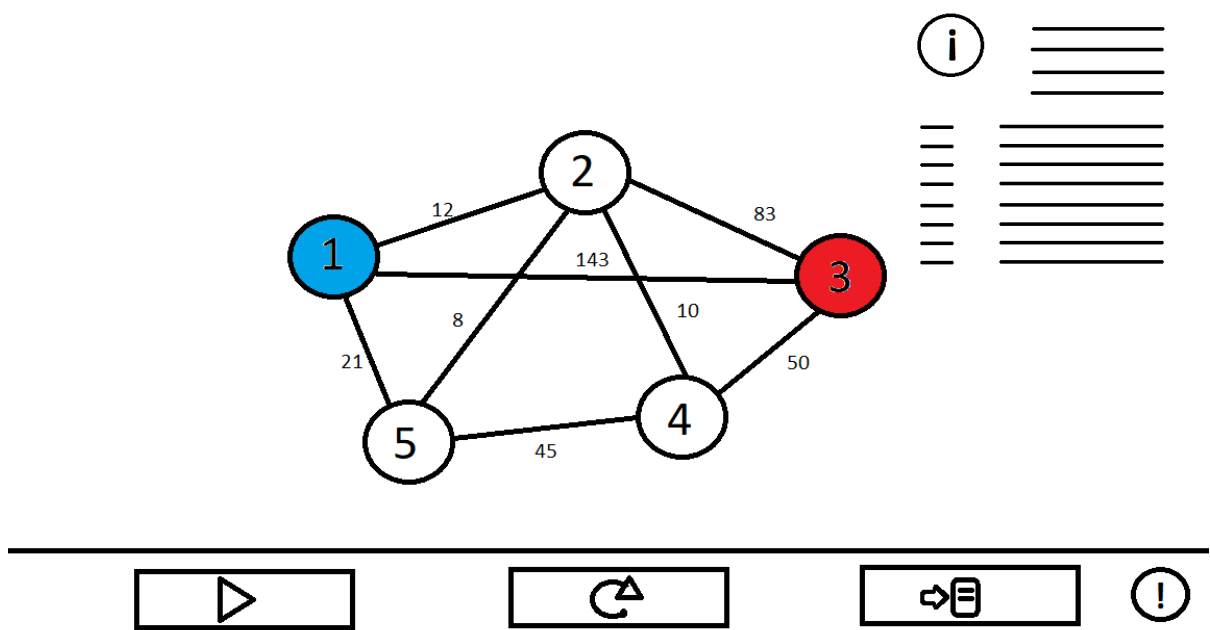


Рисунок 2 – эскиз интерфейса

2. ПЛАН РАЗРАБОТКИ И РАСПРЕДЕЛЕНИЕ РОЛЕЙ В БРИГАДЕ

2.1. План разработки

К 02.07.2020 должны быть распределены роли между членами бригады, составлена UML - диаграмма программы, а также создана директория для проекта.

К 03.07.2020 необходимо предоставить прототип приложения.

К 06.07.2020 необходимо сделать ввод графа с проверкой корректности вводимых данных, метод решения алгоритма, прорисовка графа с помощью графического интерфейса, а также добавить в отчет описание алгоритма и план тестирования. (версия 1).

К 08.07.2020 должны быть добавлены ввод из файла графа и его визуализация, также должны быть сделаны тесты для созданных структур данных и функций алгоритма согласно плану тестирования (версия 2).

К 10.07.2020 проект должен быть полностью готов, программа должна корректно собираться, в ходе сборки должны выполняться и успешно завершаться модульные тесты.

2.2. Распределение ролей в бригаде

Кобенко В.П. отвечает за разработку графического интерфейса.

Вербин К.М. отвечает за реализацию логики алгоритма.

Ефимова М.А. отвечает за тестирование и сборку приложения.

3. ОСОБЕННОСТИ РЕАЛИЗАЦИИ

3.1. Структура данных

3.2 Основные методы

3.3. Версия 0. Прототип

Проект состоит из четырех классов : DrawUtil, GraphPanel, MainWindow и Main.

Описание класса DrawUtil:

В классе имеется один метод и одно приватное поле Graphics2D – фундаментальный класс для того, чтобы представить 2-мерные формы, текст и изображения на Java (TM) платформа.

Метод `public static Color parseColor(String colorStr)` преобразует шестнадцатеричный цвет в код RGB.

Описание класса GraphPanel:

Класс наследуется от JPanel. В классе присутствует метод : `reset()`. Метод `reset` осуществляет очищение поля.

Описание класса MainWindow:

В классе MainWindow присутствуют четыре метода : `private void setGraphPanel()`, `private void setTopPanel()`, `private void setButtons()`, `private void setupIcon(JButton button, String img)`.

Метод `private void setGraphPanel()` создает холст для рисования графа, его размеры, создает кнопки и верхнюю панель.

Метод `private void setTopPanel()` создает верхнюю панель с выводом информации о создателях данного проекта.

Метод `private void setButtons()` создает управляющую панель, на которой создаются четыре кнопки : `info`, `reset`, `run` и `file`.

При нажатии на кнопку `Info` выводится информация об управлении в приложении.

При нажатии на кнопку `Reset` холст очищается, и граф можно ввести заново.

При нажатии на кнопку `Run` выводится следующий шаг работы алгоритма.

При нажатии на кнопку `File` будет осуществляться ввод из файла.

Метод `private void setupIcon(JButton button, String img)` позволяет создать иконку каждой кнопки на холсте.

Описание класса `Main`:

В классе `Main` присутствует метод `main()`, который создает окно и запускает приложение.

На рисунке 3 изображен работа прототипа приложения.

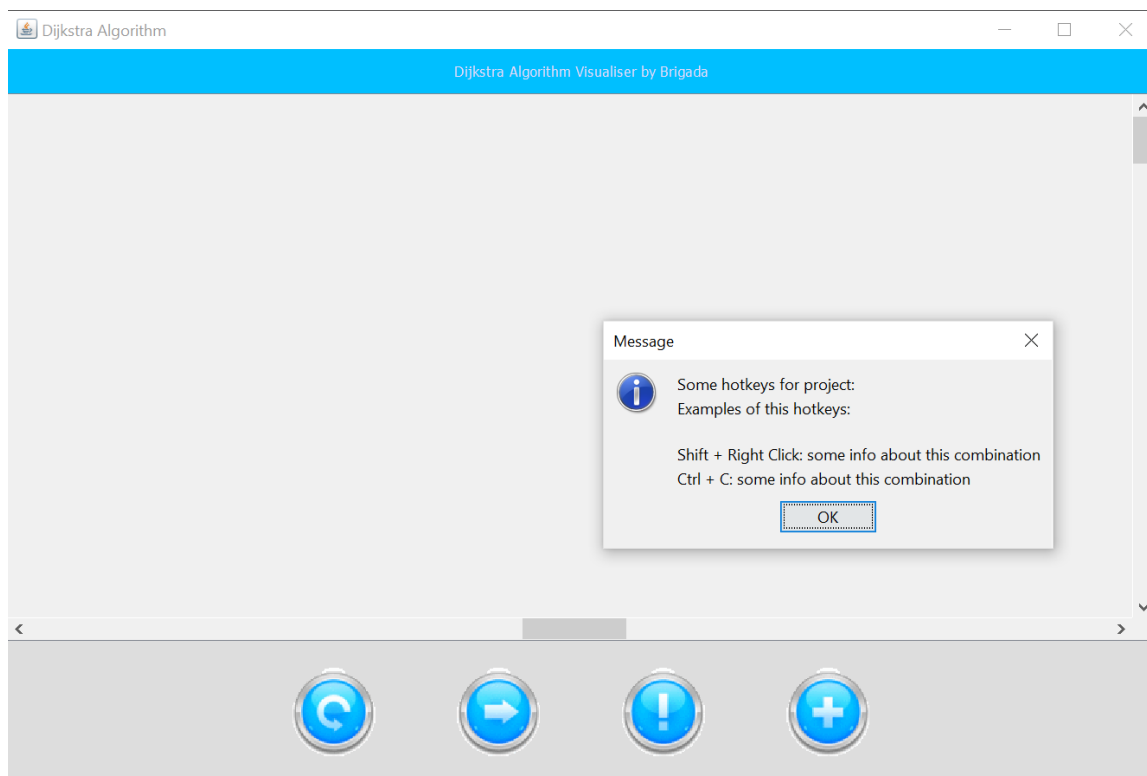


Рисунок 3 – работа прототипа приложения