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

ОТЧЕТ

по учебной практике

Тема: Кратчайшие пути в графе. Алгоритм Форда-Беллмана

Студент гр. 8383 _____ Бессуднов Г.И.

Студент гр. 8383 _____ Дейнега В.Е.

Студентка гр. 8383 _____ Кормщикова А.О.

Руководитель _____ Фирсов М.А.

Санкт-Петербург
2020

ЗАДАНИЕ
НА УЧЕБНУЮ ПРАКТИКУ

Студент Бессуднов Г.И. группы 8383

Студент Дейнега В.Е. группы 8383

Студентка Кормщикова А.О. группы 8383

Тема практики: Кратчайшие пути в графе. Алгоритм Форда-Беллмана

Задание на практику:

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

Алгоритм: Форда-Беллмана.

Сроки прохождения практики: 29.06.2020 – 12.07.2020

Дата сдачи отчета: 08.07.2020

Дата защиты отчета: 10.07.2020

Студент		Бессуднов Г.И.
Студент		Дейнега В.Е.
Студентка		Кормщикова А.О.
Руководитель		Фирсов М.А.

АННОТАЦИЯ

В данной работе по учебной практике реализуется и визуализируется пошагово алгоритм Форда-Беллмана с сопровождающими работу текстовыми пояснениями. Разработанное приложение было реализовано на языке программирования Java. Приложение оснащено графическим интерфейсом, который является ясным и удобным для пользователя. Взаимодействие с графическими элементами осуществлено с помощью мыши. Текстовые пояснения для удобства вынесены в область логгера. Также реализован ввод данных с файла. Разработка программы выполнялась итеративно, были сделаны прототип, 1-2 версия программы и финальная версия.

SUMMARY

Ford-Bellman algorithm with accompanying text explanations. The developed application was implemented in the Java programming language. The application is equipped with a graphical interface. Interaction with graphic elements is carried out using the mouse. Textual explanations for ease of removal to the logger area. Also implemented data input from a file. Program development was carried out iteratively, prototypes, 1-2 versions of the program and the final version were made.

СОДЕРЖАНИЕ

	Введение	5
1.	Требования к программе	6
1.1.	Исходные требования к программе	6
1.2.	Уточнение требований после сдачи 1-ой версии	7
1.3.	Уточнение требований после сдачи 2-ой версии	8
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.	Структуры данных	10
3.2.	Основные методы	11
4.	Тестирование	14
4.1	План тестирования	14
4.2	Тестирование графического ввода	16
4.3	Тестирование ввода с файла	18
4.4	Тестирование работы алгоритма	19
4.5	Тестирование корректного вывода логгера	20
4.6	Влияние смены режима работы на данные и работу алгоритма	22
4.7	Проверка корректной визуализаии алгоритма	23
4.8	Проверка корректности работы графического интерфейса	24
	Заключение	25
	Список использованных источников	26
	Приложение А.	27
	Приложение Б.	28

ВВЕДЕНИЕ

Целью данной практической работы является ознакомление с принципами работы в команде над приложением, ознакомление с языком программирования Java и его инструментами для работы над графическими приложениями.
Одной из задач выступает организация совместной работы с помощью системы контроля версий и разделения обязанностей разработки, а также поэтапная разработка приложения с четко выделенными версиями программы.
Другой задачей является изучение и организация алгоритма Форда-Беллмана. Данный алгоритм позволяет найти в графе кратчайшие расстояния от стартовой вершины ко всем другим, допуская наличие ребер с отрицательным весом. Алгоритм применяется в протоколе маршрутизации RIP.

1. ТРЕБОВАНИЯ К ПРОГРАММЕ

1.1. Исходные Требования к программе

1.1.1. Требования к вводу исходных данных

Входные данные будут вводится либо из файла, либо пользователем посредством щелчка мыши на холсте. Пользователь сможет создавать вершины графа, а также ребра. Вершины создаются при клике ЛКМ в области холста и удаляются при клике по ним ПКМ. Ребра создаются путем соединения существующих вершин, при добавлении выскакивает диалоговое окно, в которое пользователь вводит вес ребра, удаление ребер происходит при нажатии на них ПКМ.

Формат данных в файле: Данные в файле подаются в следующем виде: сначала записывается количество ребер n и стартовая вершина, затем последовательно идут n строк имеющих формат:

"Начало(число)" "Конец(число)" "Вес(число)"

Пример:

5

1

1 1 2

1 2 3

3 3 -1

1 2 -4

5 3 4

1.1.2. Требования к визуализации

Прототип интерфейса программы представлен в приложении А. Будут реализованы два Layout’а, которые визуализируют два режима работы: режим редактирования (Edit mode), режим работы алгоритма (Watch mode);. Пользователь видит граф на холсте, на ребрах будут отмечены веса, у вершин будет метка о дистанции до них. В Watch mode - пользователь увидит активное

ребро, конечную вершину этого ребра, новую метку, которая, возможно, будет записана. Все активные элементы будут выделены иным цветом

1.1.3. Требования к работе режимов.

Переключение между режимами происходит посредством нажатия соответствующей кнопки. Edit mode - предоставляет возможность редактировать граф. Watch mode - позволяет сделать шаг алгоритма, либо же просмотреть полностью работу алгоритма.

1.1.4. Требования к логированию

Логи будут зависит от режима работы. В Edit Mode выводится информация о добавленных или удаленных ребрах или вершинах. В Watch Mode выводится информация о шаге алгоритма: выводится просматриваемое ребро, для него вычисляется новое предполагаемое расстояние - выводится старое значение и новое, пишется сообщение о том, произошла ли замена. При завершении одного этапа релаксации будет выводится информация о завершении этого этапа. Также будет выводится сообщение о начале работы алгоритма, инициализации алгоритма (выводится стартовая вершина(номер) ее метка - "0" и сообщение, в котором сказано, что все остальные вершины имеют метку "бесконечность"), о завершении работы алгоритма (выводится список вершин и расстояние до них).

UML-диаграмма проекта представлена в приложении Б.

1.2. Уточнение требований после сдачи 1-ой версии

1.2.1. Обозначение вершин буквами, в т.ч. в формате файлов. Новый формат данных для ввода:

"Начало(буква)" "Конец(буква)" "Вес(число)"

Пример:

5

a

a a 2

a b 3

c d -1

d a -4

b c 4

1.2.2. Рёбра должны быть достаточно толстыми, чтобы щелчок по ним не вызывал затруднений.

1.2.3. Вес ребра должен печататься максимально близко к линии ребра и не должен выводиться ближе к другому ребру.

1.2.4. Сообщения логгера должны быть доступны для копирования.

1.2.5. В логгере должно отмечаться, когда все рёбра перебраны и запускается новый круг.

1.2.6. В логгере должна выводиться причина завершения работы алгоритма.

1.3. Уточнение требований после сдачи 2-ой версии

1.3.1. Полупрозрачность весов рёбер - это очень хорошо, но, к сожалению, когда ребро проходит по цифрам, это мешает их видеть. С этим надо что-то сделать (возможное решение: делать число видимым на фоне ребра при наведении курсора).

1.3.2. Возможность остановить процесс после нажатия на "Run full". - Определять, какие вершины входят в отрицательные циклы.

1.3.3. Добавить кнопку "1 cycle", по нажатию на которой будет мгновенно выполняться 1 цикл шагов.

1.3.4. Если метка вершины в результате выполнения шага изменилась, то она должна становиться красной и оставаться красной до начала следующего шага (если метка не изменилась, то оставить прежнее поведение с маленькой задержкой перед почернением).

1.3.4. Оптимизировать алгоритм: если в результате выполнения цикла не было изменений, то алгоритм должен завершать работу.

1.3.6. Для вершин с меткой inf в логгере должен выводиться соответствующий результат, а не 0.

2. ПЛАН РАЗРАБОТКИ И РАСПРЕДЕЛЕНИЕ РОЛЕЙ В БРИГАДЕ

2.1. План разработки

1. Прототип, 3 июля. Включает в себя:
   - Необходимые интерфейсы программы
   - Элементы пользовательского интерфейса. (Визуализация)
   - Возможность переключения между режимами работы
2. Версия 1, 5 июля. Включает в себя:
   - Возможность создать граф
   - Рабочий логгер
   - Рабочий алгоритм, без пошаговой визуализации.
3. Версия 2, 7 июля. Включает в себя:
   - Визуализация алгоритма
   - Считывания из файла
   - Отполированный логгер

2.2. Распределение ролей в бригаде

Бессуднов Глеб - реализует редактор графа

Дейнега Виктора - реализует GUI

Кормщикова Арина - реализует алгоритм

3. ОСОБЕННОСТИ РЕАЛИЗАЦИИ

3.1. Структуры данных

3.1.1. Структура данных Graph

Класс для хранения информации о графе. Имеет свой собственный интерфейс IGraph, через который с ней взаимодействуют другие части программы и используется в классах GraphEditor и Algorithm (каждый экземпляр класса хранит свою копию).

Класс Graph содержит следующие поля:

public Map<Vertex, ArrayList<Edge> > graph; - контейнер, представляющий из себя список смежности. Поле Key - вершина графа, Value - список ребер, исходящих из этой вершины.

public Vertex startVertex; - поле в котором хранится стартовая вершина для алгоритма

public int countOfVertex - количество вершин в графе

public int countOfEdge - количество ребер в графе

3.1.2. Структура данных Vertex

Класс для хранения информации о вершине. Содержит следующие поля:

public int number - номер вершины.

public int distance - дистанция до вершины

public boolean isStart - является ли вершина стартовой

public boolean isCheck - было ли изменении дистанции с метки "бесконечность"

public String name - имя вершины

3.1.3. Структура данных Edge

Класс для хранения информации о ребре. Содержит следующие поля:

public Vertex start - начальная вершина

public Vertex end - конечная вершина

public int weight - вес ребра

3.1.4. Структура данных AlgorithmMessage

Класс хранящий в себе информацию о шаге алгоритма.Содержит следующие поля:

private String message - сообщение о работе алгоритма.

private Edge viewingEdge - рассматриваемое на шаге алгоритма ребро

private Vertex changeV - конечная вершина ребра (для которой изменяется дистанция)

private Vertex startV - начальная вершина ребра

private boolean isFinish - поле, отображающее закончил ли алгоритм работу

private boolean isEndOfCycle - поле, отображающее закончился ли текущий цикл алгоритма

3.2. Основные методы

3.2.1. Интерфейс IGraphEditor

public abstract void setEditState(boolean isEditState); - функция для установки состояния редактора. Принимает переменную boolean isEditState, если она равна true, то редактор переходит в режим редактирвоания, в ином случае он переходит в режим просмотра.

public abstract IGraph getGraph(); - возвращает объект типа IGraph, который является редактируемым графом.

public abstract void setCurrentEdge(Edge e); - функция, которая устанавливает и корректно отображает текущее просматриваемое ребро при работе алгоритма. Принимает объект e типа Edge - ребро, визуальное представление которого необходимо обновить.

public abstract void setCurrentVertex(Vertex v); - функция, которая устанавливает и корректно отображает текущую просматриваемую вершину при работе алгоритма. Принимает объект v типа Vertex - вершину, визуальное представление которой необходимо обновить.

public abstract void clearEditor(); - функция для очистки редактора от всего содержимого в нем.

public abstract void loadGraph(Graph graph); - функция для загрузки графа в редактор с последующим созданием его визуального пердставления. Принимает объект graph - граф, который необходимо построить.

public abstract void rerunEditor(); - функция для перезапуска редактора. Возвращает редактор в его вид, перед началом алгоритма. Не удаляет его содержимое.

3.2.2. Интерфейс IGraph

public abstract void addVertex(Vertex v); - метод, принимающий вершину Vertex v. Добавляет данную вершину в граф.

public abstract void addEdge(Edge e) - метод, принимающий ребро Edge e. Добавляет данное ребро в граф.

public abstract void deleteEdge(Edge e) -метод, принимающий ребро Edge e, удаляет данное ребро из графа.

public abstract void deleteVertex(Vertex v) - метод, принимающий вершину Vertex v, удаляет данную вершину из графа.

public abstract void setStartVertex(Vertex v) - метод, принимающий вершину Vertex v, устанавливает данную вершину как начальную для алгоритма

3.2.2. Интерфейс IAlgorithm

public abstract AlgorithmMessage stepForward() - метод, реализующий один шаг алгоритма (просмотр одного ребра). Во время своей работы создает AlgorithmMessage и возвращает его.

public abstract void initAlgorithm(Graph g) - метод инициализации, принимает граф Graph g. Ставит алгоритм в начальное состояние

3.2.3. Методы fileReader

public Graph readFromFile() - метод, в котором происходит считывание данных из файла формата .txt. Возвращает null - если файл/данные в файле

некорректны, при корректных данных - возвращает Grahp построенные по этим данным.

3.2.4. Интерфейс ILogger

public abstract void logEvent(String message); - метод, принимающий строку и выводящий ее на экран.

public abstract void logEvent(AlgorithmMessage message); - метод, принимающий AlgorithmMessage message и выводящий поле типа String этого класса.

public abstract void clear(); - метод, очищающий контейнер, отображающийся в качестве логгера.

public abstract String prepare(String message); - метод, подготавливающий строку логга, добавляя символ переноса строки так, чтобы лог корректно отображался на экане

4. ТЕСТИРОВАНИЕ

4.1. План тестирования

1. Что надо тестировать:
    1. Графический ввод.
    2. Ввод с файла.
    3. Работа алгоритма.
    4. Тестирование корректного вывода логгера
    5. Влияние смены режима работы на данные и работу алгоритма.
    6. Проверка корректной визуализации алгоритма.
    7. Проверка корректности работы графического интерфейса.
2. Как будем тестировать:
    1. Тесты вручную: добавление вершины, ребра, удаление вершины, ребра.
    2. Тесты на ввод корректных файлов, тесты на ввод некорректных файлов
    3. Тесты на графе с одной вершины, несвязный граф, пустой граф, правильность решения на корректных графах
    4. Проверка всех возможных комбинаций клавиш на корректность ответа логера на них. Проверка на переполняемость логгера. Проверка корректности переключения логера между двумя режимами работы программы. (У Edit mode и Watch mode логирование не пересекается).
    5. Проверка корректного сброса прогресса алгоритма при переключении Watch=>Edit=>Watch.
    6. Ручное тестирование с пошаговым прогоном алгоритма на различных графах: с одной вершиной, несвязном графе, пустом графе, корректном графе.
    7. Тесты на работу взаимодействия пользователя и интерфейса: каждая кнопка корректно обрабатывает нажатие. Проверка на блокировку недоступных в разных режимах кнопок.
3. Когда будем тестировать:
    1. Между 1 и 2 версиями программы

2. Между 1 и 2 версиями программы
3. Между прототипом и 1 версией программы. После 2й версии программы.
4. Между прототипом и 1 версией программы. После 2й версии программы.
5. Между 1 и 2 версиями программы.
6. Между прототипом и 1 версией программы. После 2й версии программы.
7. До и после 2й версии.

4.2. Тестирование графического ввода

Были проведены теста на добавление вершины, ребра, удаление вершины ребра

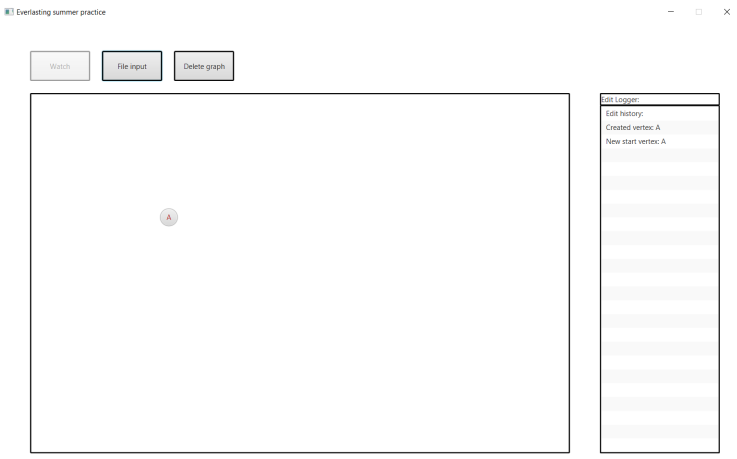

Рисунок 1 - Тест №1. Добавление вершин и ребра.

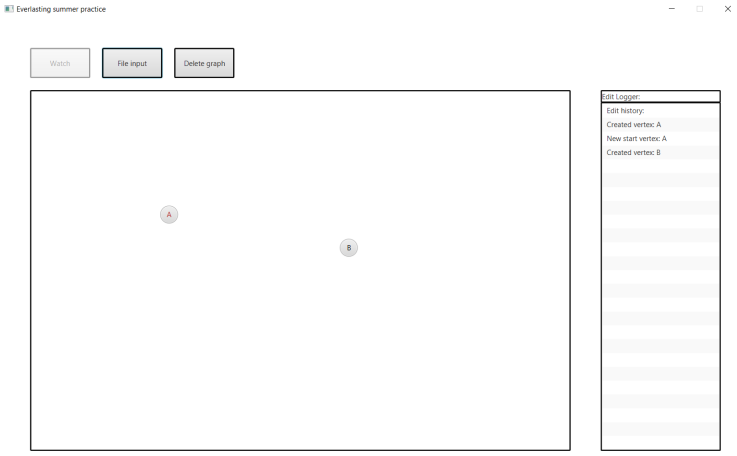

Рисунок 2 - Тест №1. Добавление вершин и ребра.

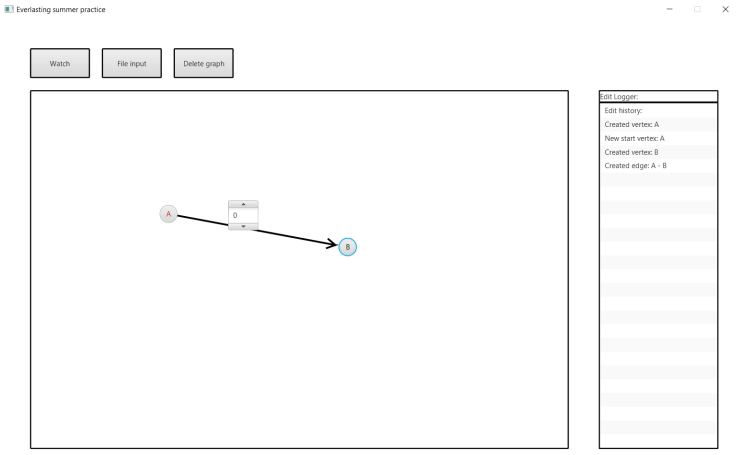

Рисунок 3 - Тест №1. Добавление вершин и ребра.

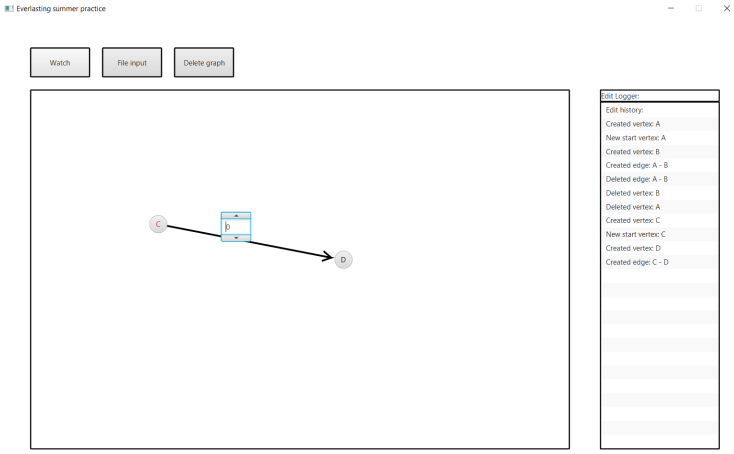

Рисунок 4 - Тест №2. Удаление вершин.

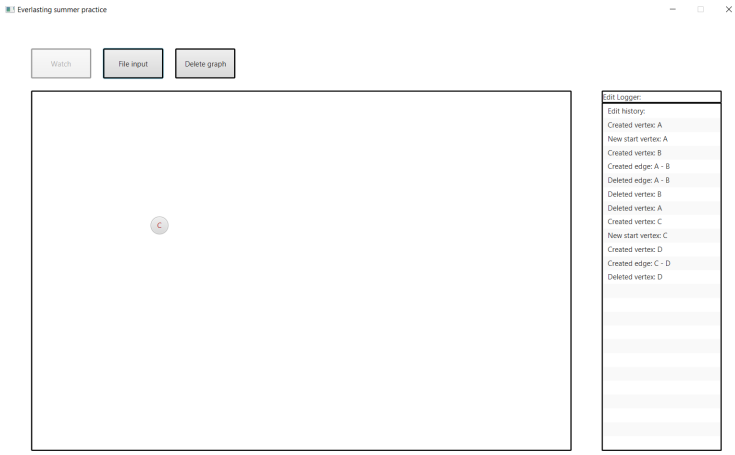

Рисунок 5 - Тест №2. Удаление вершин.

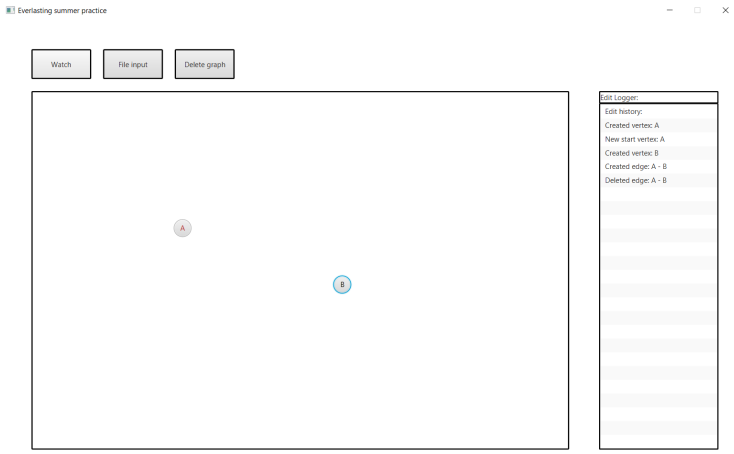

Рисунок 6 - Тест №2. Удаление ребра.

4.3. Тестирование ввода с файла

Были проведены тесты на ввод корректных файлов и некорректных.

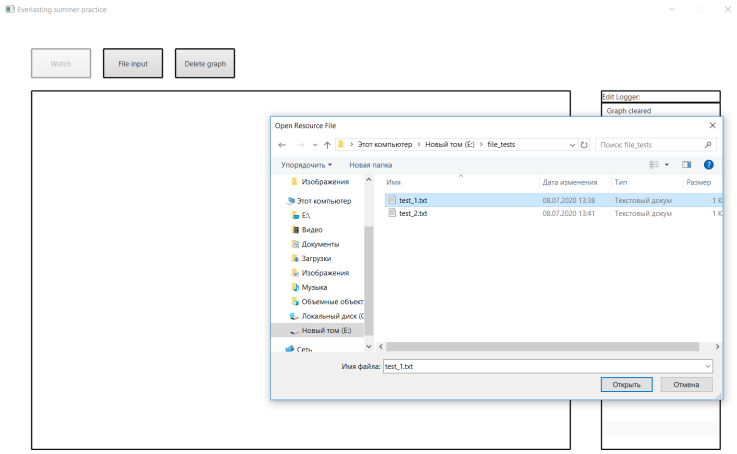

Рисунок 7 - Тест №3. Выбор файла.

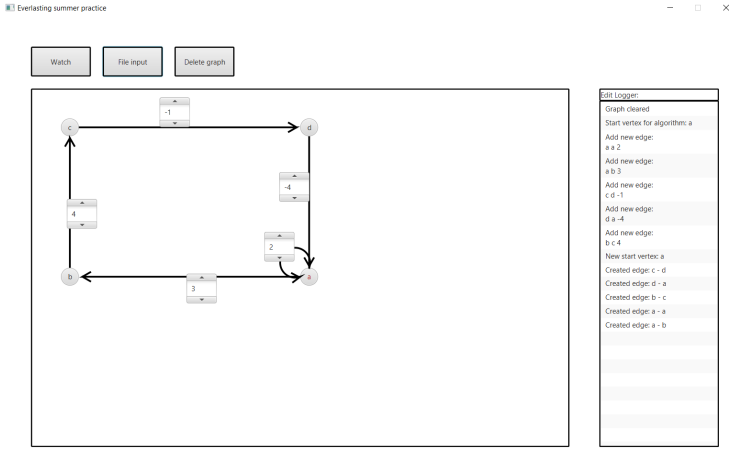

Рисунок 8 - Тест №3. Построение графа по данным из файла.

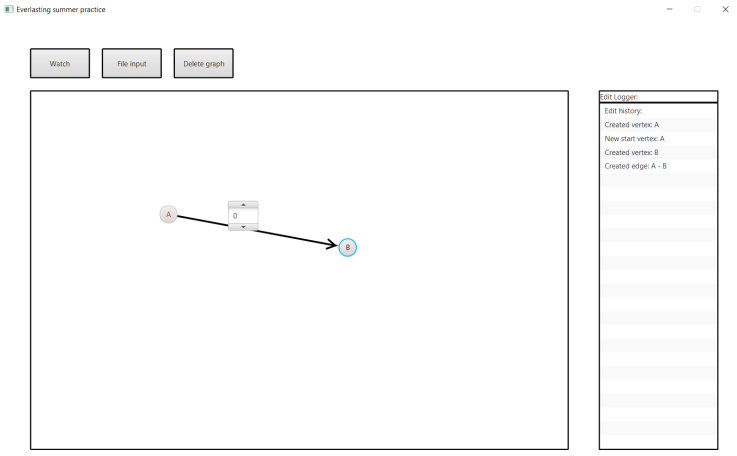

Рисунок 9 - Тест №4. Ввод с файла с некорректными данными.

4.4. Тестирование работы алгоритма

Были проведены тесты на различных графах. Для теста 7 лог приведен в приложении Б.

Рисунок 10 - Тест №5. Алгоритм на пустом графе

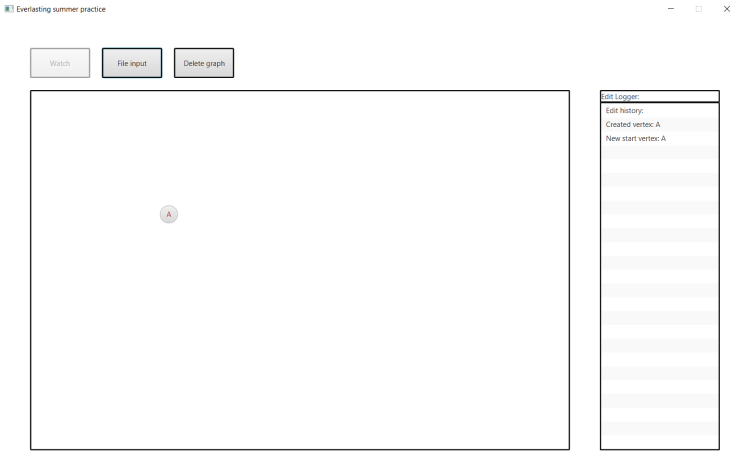

Рисунок 11 - Тест №6. Алгоритм на графе из одной вершины.

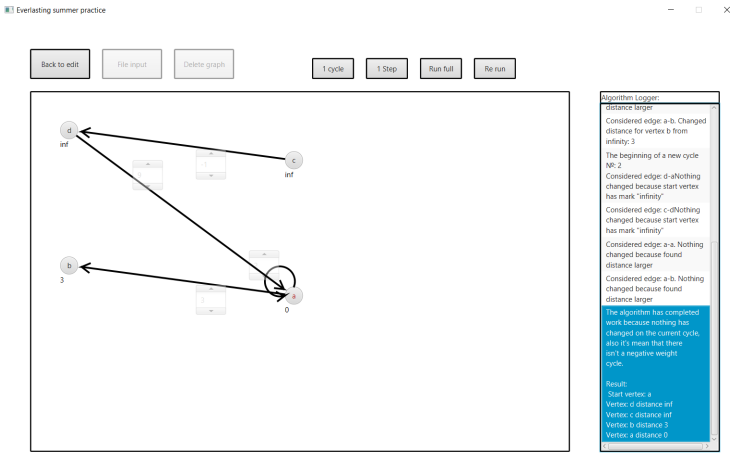

Рисунок 12 - Тест №7. Алгоритм на графе.

4.5. Тестирование корректного вывода логгера

Были проведены тесты на перебор всех возможных комбинаций клавиш и корректность ответа логера на них .

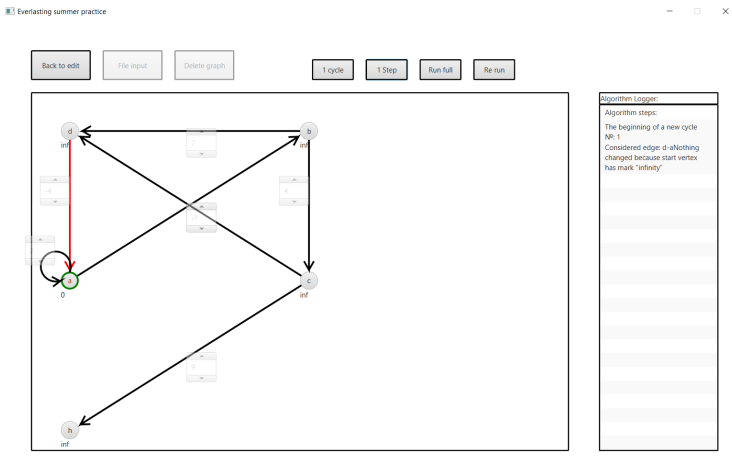

Рисунок 13 - Тест №8. Логгер - 1 шаг алгоритма

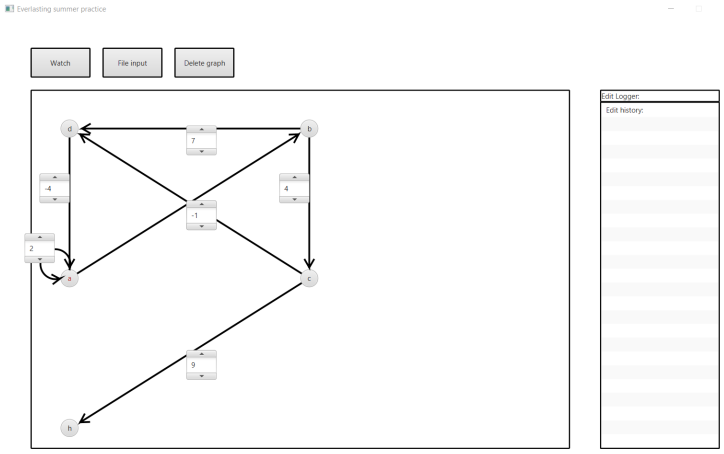

Рисунок 14 - Тест №9. Логгер - Переход в Edit mode.

Рисунок 15 - Тест №10. Логгер - Кнопка Delete graph.

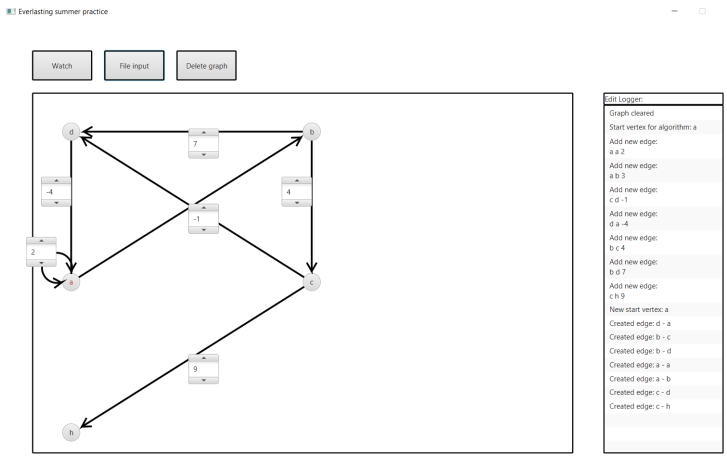

Рисунок 16 - Тест №11. Логгер - Ввод с файла

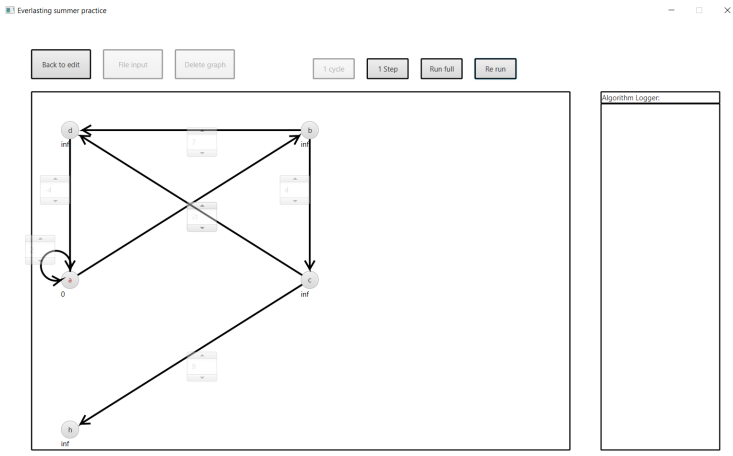

Рисунок 17 - Тест №12. Логгер - Кнопка Re run.

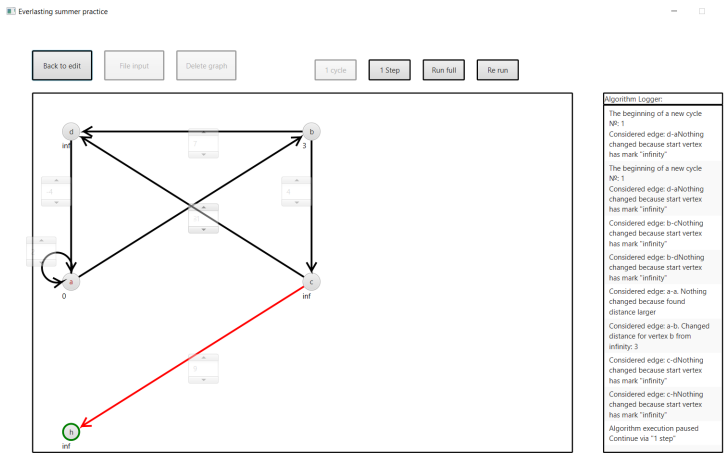

Рисунок 18 - Тест №13. Логгер - Кнопка Stop (при режиме Run full).

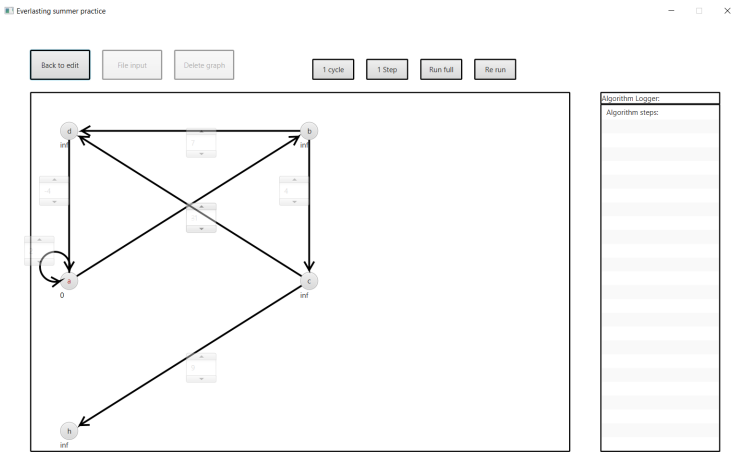

Рисунок 19 - Тест №14. Логгер - переход в Watch mode

4.6. Влияние смены режима работы на данные и работу алгоритма.

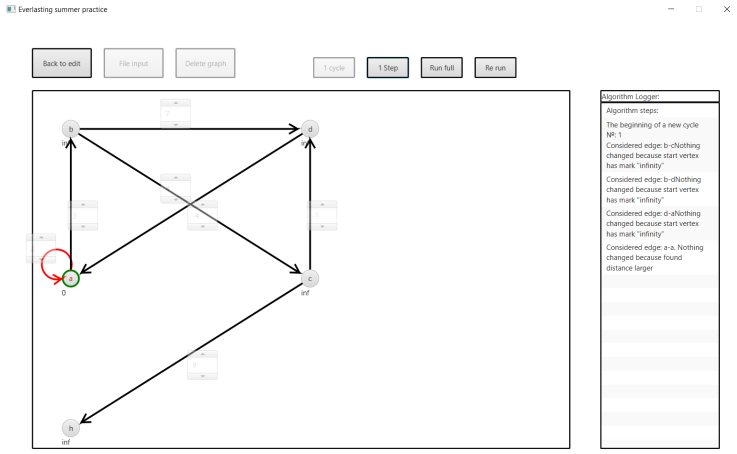

Рисунок 20 - Тест №15. Алгоритм в состоянии Watch mode.

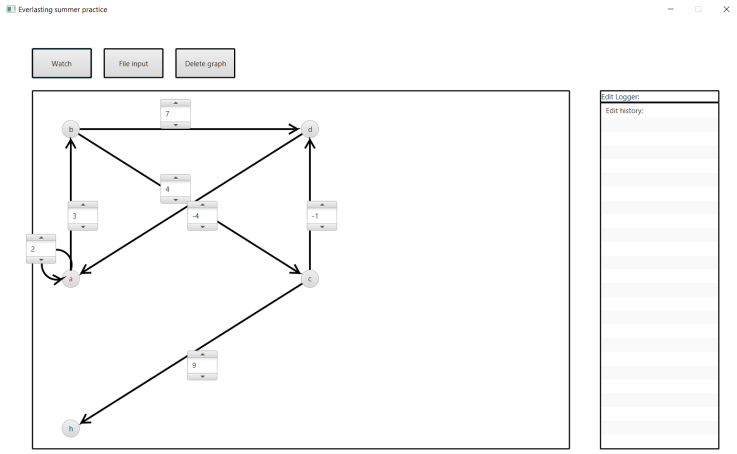

Рисунок 21 - Тест №15. Алгоритм перешел в состояние Edit mode.

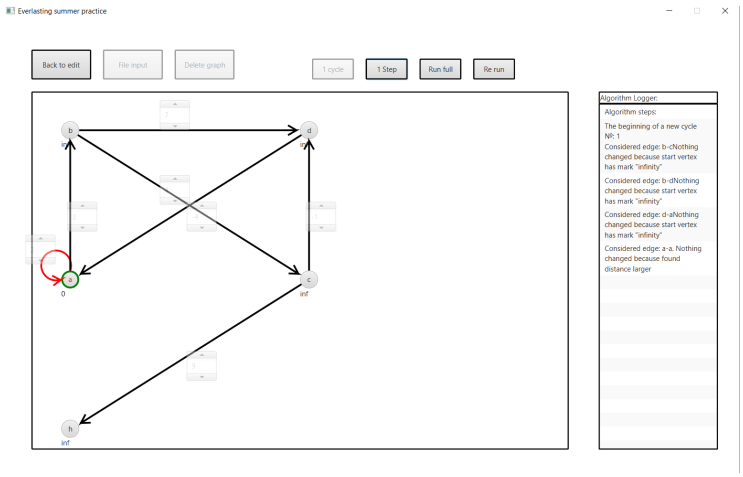

Рисунок 22 - Тест №15. Алгоритм перешел обратно в состояние Watch mode

4.7. Проверка корректной визуализации алгоритма.

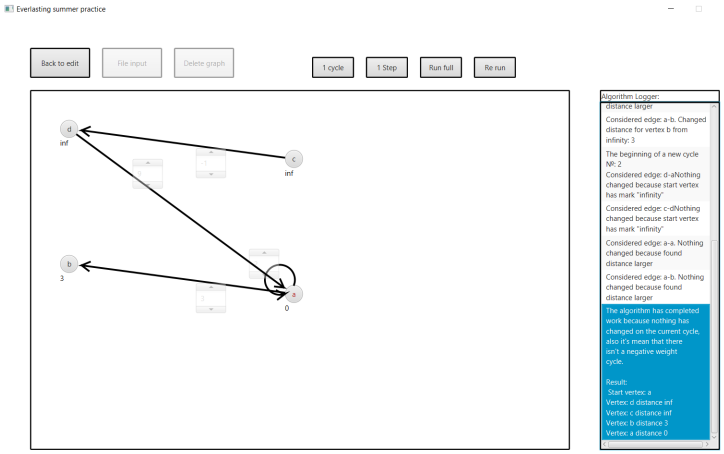

Рисунок 23 - Тест №16. Алгоритм завершил работу.

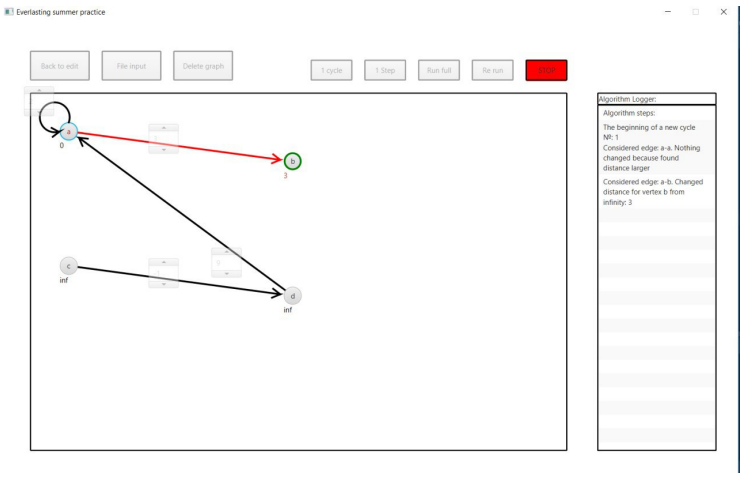

Рисунок 24 - Тест №17. Выделение текущего ребра во время работы алгоритма.

4.8. Проверка корректности работы графического интерфейса

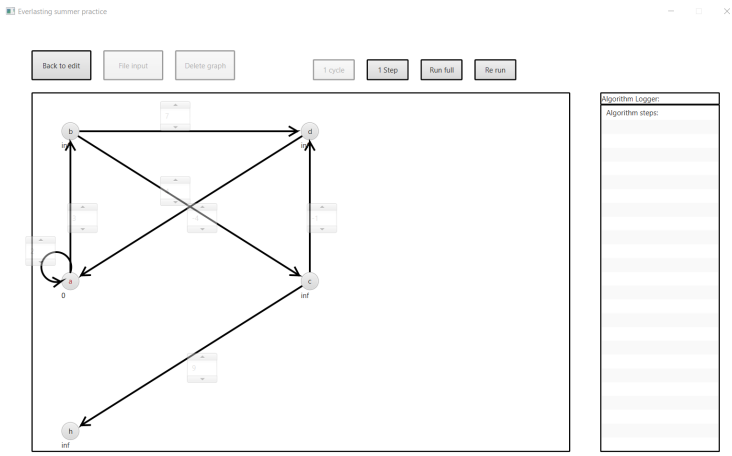

Рисунок 25 - Тест №18. Блокировка кнопок в Watch mode

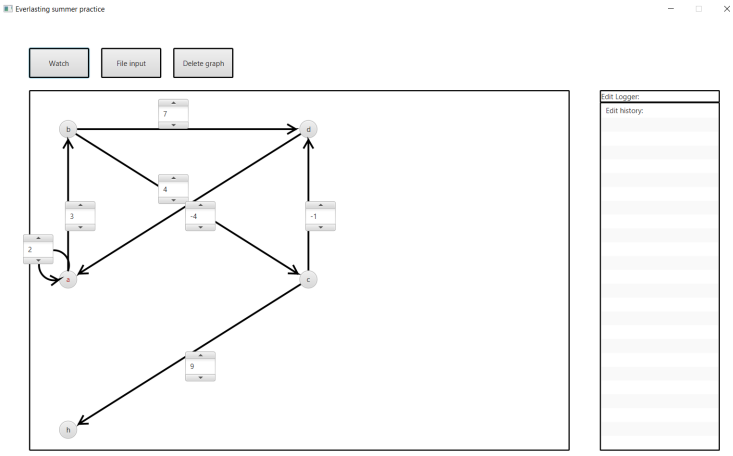

Рисунок 26 - Тест №18. Активные кнопки в Edit mode.

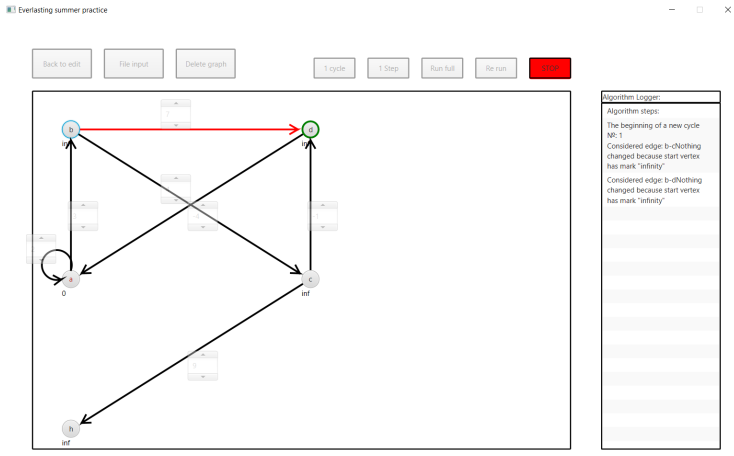

Рисунок 27 - Тест №18. Активные/неактивные кнопки в Run full.

ЗАКЛЮЧЕНИЕ

В ходе выполнения практической работы было успешно реализовано приложение, написанное на языке Java с использованием его инструментов для работы с GUI, которое визуально демонстрирует алгоритм Форда-Беллмана. Так же всеми участниками группы был получен опыт работы в команде с использованием системы контроля версий и распределением обязанностей. Приложение было сделано поэтапно, в четыре версии. Каждая версия является самостоятельной рабочей программой.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. A computer science portal for geeks// www.geeksforgeeks.org. URL: https://www.geeksforgeeks.org/bellman-ford-algorithm-simple-implementation/ (дата обращения: 30.06.2020).

2. Сайт о программировании // metanit.com. URL: https://metanit.com/java/ (дата обращения: 30.06.2020).

3. Java documentation// docs.oracle.com. URL: https://docs.oracle.com/javase/8/javafx/api/index.html (дата обращения: 30.06.2020).

ПРИЛОЖЕНИЕ А
ССЫЛКА НА РЕПОЗИТОРИЙ С ИСХОДНЫМ КОДОМ

https://github.com/GlebDot/SummerPractice/tree/master/Code/defaultArtifact/src/main/java

ПРИЛОЖЕНИЕ Б
ТЕСТ 7 ЛОГИРОВАНИЕ

Algorithm steps:

The beginning of a new cycle
є: 1
Considered edge: d-a Nothing
changed because start vertex
has mark "infinity"

Considered edge: c-d Nothing
changed because start vertex
has mark "infinity"

Considered edge: a-a. Nothing
changed because found
distance larger

Considered edge: a-b. Changed
distance for vertex b from
infinity: 3

The beginning of a new cycle
є: 2
Considered edge: d-a Nothing
changed because start vertex
has mark "infinity"

Considered edge: c-d Nothing
changed because start vertex

has mark "infinity"

Considered edge: a-a. Nothing
changed because found
distance larger

Considered edge: a-b. Nothing
changed because found
distance larger

The algorithm has completed
work because nothing has
changed on the current cycle

also it's mean that there
isn't a negative weight
cycle.

Result:
Start vertex: a
Vertex: d distance inf
Vertex: c distance inf
Vertex: b distance 3
Vertex: a distance 0