**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра МО ЭВМ**

### Отчет

### по учебной практике

**«Визуализация алгоритмов на графах на Java»**

Студент гр. 6383 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Быков И. В.

Студент гр. 6383 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Лавренкова Е. Л.

Студент гр. 6381 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Кухарев М. А.

Руководитель \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Жангиров Т.?.

Санкт-Петербург

2018

**ЗАДАНИЕ**

**на учебную практику**

|  |  |  |
| --- | --- | --- |
| Студент Быков И. В. группы 6383 | | |
| Студентка Лавренкова Е. Л. группы 6383 | | |
| Студент Кухарев М.А. группы 6381  Тема практики: Визуализация алгоритмов на графах на Java | | |
| Задание на практику:  Командная итеративная разработка визуализатора алгоритма на Java с графическим интерфейсом.  Алгоритм: Форда – Беллмана. | | |
| Сроки прохождения практики: 27.06.2018 – 10.07.2017 | | |
| Дата сдачи отчета: ??.07.2018 | | |
| Дата защиты отчета: ??.07.2018 | | |
|  | | |
| Студент |  | Быков И. В. |
| Студентка |  | Лавренкова Е. Л. |
| Студент |  | Кухарев М. А. |
| Руководитель |  | Жангиров Т.?. |

**АННОТАЦИЯ**

Целью данной работы является разработка визуализатора алгоритма Форда — Беллмана на языке Java с использованием графического интерфейса. Данный алгоритм производит поиск кратчайшего пути во взвешенном графе с возможными ребрами отрицательного веса. В результате выполнения работы программы, пользователь получает граф, на котором отображен кратчайший путь от одной вершины к другой.

**annotation**

The purpose of this work is to develop a visualizer of the Bellman — Ford algorithm in the Java language using a graphical interface. This algorithm searches for the shortest path in a weighted graph with possible edges of negative weight. As a result of the execution of the program, the user receives a graph on which the shortest path from one vertex to another is displayed.

**Содержание**

[Отчет 1](#_Toc486917567)

[по учебной практике 1](#_Toc486917568)

[1. ИСХОДНОЕ ЗАДАНИЕ 5](#_Toc486917569)

[2. ОБЪЯСНЕНИЕ АЛГОРИТМА 5](#_Toc486917570)

[2.1. Теоретическая справка 5](#_Toc486917571)

[2.2.Описание алгоритма 5](#_Toc486917572)

[3. ФОРМАЛЬНАЯ ПОСТАНОВКА ЗАДАЧИ 6](#_Toc486917573)

[4. СПЕЦИФИКАЦИЯ 6](#_Toc486917574)

[4.1. Исходные Требования к программе 6](#_Toc486917575)

[4.2. Графический интерфейс 7](#_Toc486917576)

[4.3. Ход работы 9](#_Toc486917577)

[5. План разработки и распределение ролей в бригаде](#_Toc486917578) 11

[5.1. План разработки](#_Toc486917579) 11

[5.2. Распределение ролей в бригаде](#_Toc486917580) 11

[6. ТЕСТИРОВАНИЕ ПРОГРАММЫ](#_Toc486917581) 12

[6.1. Тестирование работы программы при вводе из файла in.txt:](#_Toc486917582) 12

[6.2. Тестирование работы программы с помощью случайной генерации.](#_Toc486917583) 12

[7. ЗАКЛЮЧЕНИЕ](#_Toc486917584) 12

[8. ПРИЛОЖЕНИЕ А. ИСХОДНЫЙ КОД](#_Toc486917585) 12

# 1. **ИСХОДНОЕ ЗАДАНИЕ**

В данной практической работе требуется написать программу на языке JAVA, выполняющую поиск Кратчайших путей в графе с помощью алгоритма Беллмана — Форда и выполнить визуализацию алгоритма.

# 2. **ОБЪЯСНЕНИЕ АЛГОРИТМА**

## 2.1. **Теоретическая справка**

Алгоритм Беллмана — Форда — алгоритм поиска кратчайшего пути во взвешенном графе. За время O (|V| × |E|) алгоритм находит кратчайшие пути от одной вершины графа до всех остальных. В отличие от алгоритма Дейкстры, алгоритм Беллмана-Форда допускает рёбра с отрицательным весом. Предложен независимо Ричардом Беллманом и Лестером Фордом.

## 2.2. **Описание алгоритма**

Мы считаем, что граф не содержит цикла отрицательного веса. Заведём массив расстояний , который после отработки алгоритма будет содержать ответ на задачу. В начале работы мы заполняем его следующим образом:  , а все остальные элементы   равны бесконечности ∞.

Сам алгоритм Форда-Беллмана представляет из себя несколько фаз. На каждой фазе просматриваются все рёбра графа, и алгоритм пытается произвести **релаксацию** (relax, ослабление) вдоль каждого ребра  стоимости c. Релаксация вдоль ребра — это попытка улучшить значение   значением . Фактически это значит, что мы пытаемся улучшить ответ для вершины , пользуясь ребром  и текущим ответом для вершины .

Утверждается, что достаточно  фазы алгоритма, чтобы корректно посчитать длины всех кратчайших путей в графе (повторимся, мы считаем, что циклы отрицательного веса отсутствуют). Для недостижимых вершин расстояние  останется равным бесконечности ∞.

Алгоритм Беллмана-Форда позволяет очень просто определить, существует ли в графе отрицательный цикл, достижимый из вершины. Достаточно произвести внешнюю итерацию цикла не {\displaystyle |V|-1}, a ровно {\displaystyle |V|} раз. Если при исполнении последней итерации длина кратчайшего пути до какой-либо вершины строго уменьшилась, то в графе есть отрицательный цикл, достижимый из данной вершины. На основе этого можно предложить следующую оптимизацию: отслеживать изменения в графе и, как только они закончатся, сделать выход из цикла (дальнейшие итерации будут бессмысленны).

# 3. **ФОРМАЛЬНАЯ ПОСТАНОВКА ЗАДАЧИ**

Задан исходный ориентированный граф G = (V, E), где V – количество вершин в графе, для данного графа, в котором каждая вершина пронумерована от 0 до V. Необходимо реализовать алгоритм Беллмана — Форда для данного графа.

Визуализацию необходимо проделать для небольших по размерам графов для того, чтобы убедится в правильности визуализации алгоритма. После выполнения алгоритма будет получен минимальный путь, такой, чтобы пройти из стартовой точки в конечную с минимальными затратами по стоимости.

# 4. **СПЕЦИФИКАЦИЯ**

## 4.1. **Исходные Требования к программе**

#### **4.1.1.Требования к вводу исходных данных.**

Программа предоставляет пользователю графический интерфейс. Входные данные считываются либо из файла, либо с помощью случайной генерации графа (критериями построения графа будут являться количество вершин и количество ребер, исходящих из одной вершины). Тестовый файл содержит количество вершин, количество ребер, вершины из которых исходят ребра, вершины в которые входят ребра и веса ребер графа. После запуска программы, пользователь наблюдает меню программы, в котором он может считать граф из файла или сгенерировать граф самостоятельно.

#### **4.1.2.Требования к визуализации.**

Визуализация должна представлять собой окно с меню. Граф можно либо считать из файла in.txt, при нажатии кнопки «Read from the file», либо сгенерировать, при нажатии «Generate graph».

Эскиз окна с меню выглядит следующим образом (рис.1, 2):

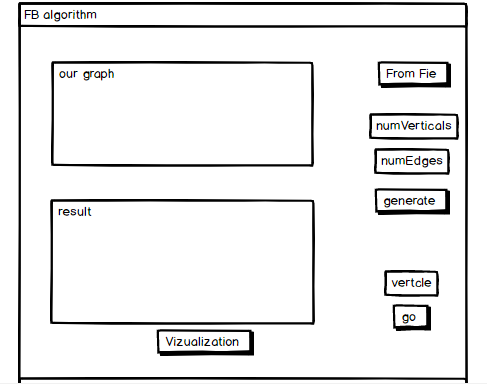


Рисунок 1

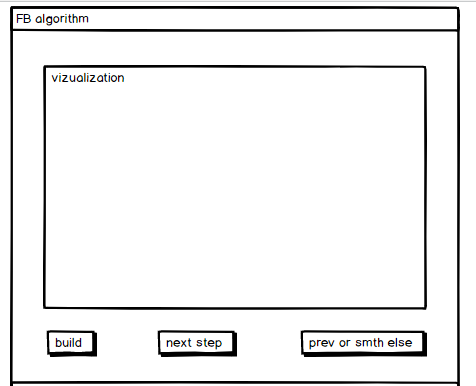


Рисунок 2

## 

## 4.2. **Графический интерфейс**

При выборе генерации графа, пользователя просят ввести количество вершин в графе (до 30) и количество ребер, исходящих из одной вершины (не более 10). При создании графа, считанного из текстового файла, расположение вершин задаются автоматически, ребра – создаются между i-ой вершиной и j-ой. Веса рёбер указываются пользователем, непосредственно в самом текстовом файле. При создании сгенерированного графа расположение вершин, ребер и веса ребер задаются автоматически.

Пользователю, после построения графа, предоставится выбор вершины. Ему необходимо ввести вершину графа в текстовое поле и нажать кнопку «Start the algorithm».

Затем к созданному графу применяется алгоритм Беллмана — Форда. Пользователь сможет просмотреть работу алгоритма в текстовом поле и запустить визуализацию работы алгоритма, нажав «Visualization of the algorithm’s work». Далее пользователю предлагается ввести конечную вершину графа с помощью текстового поля и кнопки «Path».

Пользователь может сделать выбор, в каком виде получить результат выполнения алгоритма: в виде конечного результата или пошагового выполнения.

Выходными данными является визуализированный алгоритм: построенный кратчайший путь между начальной и конечной вершинами.

Кратчайший путь в итоговом графе выделяется красным цветом. Ребро графа, которое в ходе выполнения подверглось релаксации, выделяется зелёным цветом, а ребро, которое не подверглось релаксации – синим. Если до точки, в которую входит ребро, нет пути, то оно выделяется оранжевым цветом. Программа реализована на языке Java с использованием Фреймворка JavaFX. Далее представлен план Меню:

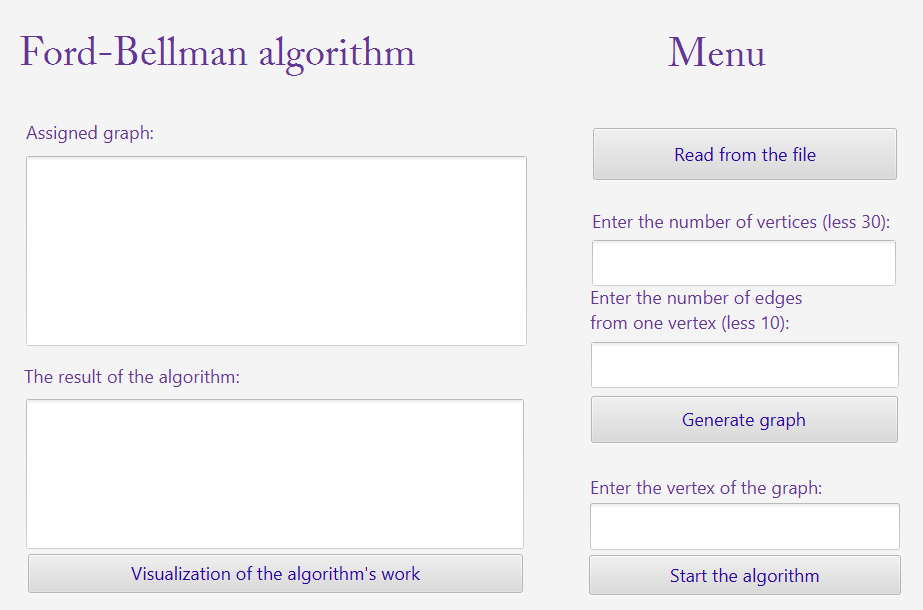


Рисунок 3

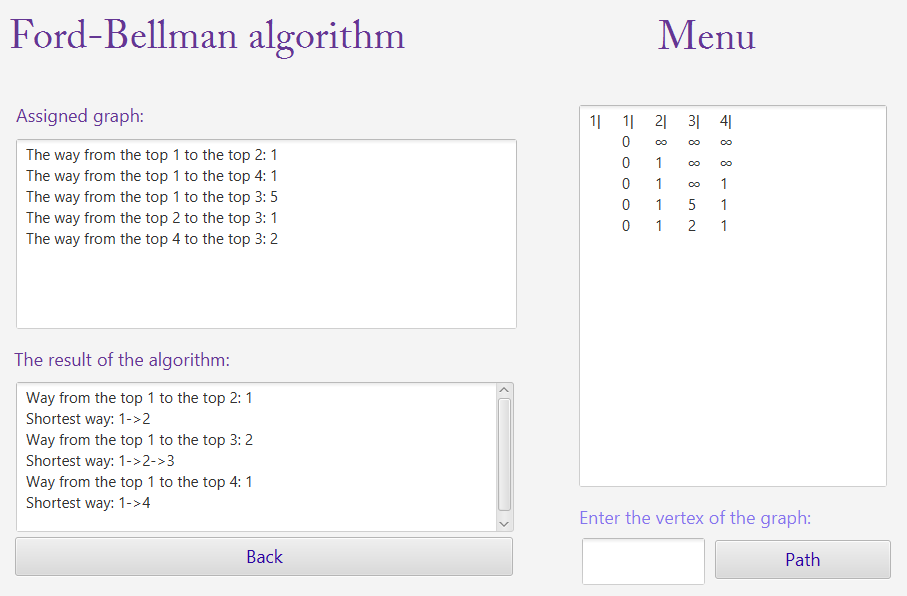


Рисунок 4

Далее представлен план дочернего окна.



Рисунок 5

4.3.Ход работы

Ход работы над разработкой программы состоит из четырёх частей:   
*1) Реализация графа.*

Граф определяется классом Graph, который состоит из класса ElementGraphWay и класса WaysPoint.   
Класс ElementGraphWay содержит поля: вершина-исток, вершина-финиш и вес ребра. Класс WaysPoint содержит координаты и имена вершин. Таким образом, граф представляется в виде списка смежности.  
*2) Реализация алгоритма Беллмана — Форда.*

Для реализации алгоритма Беллмана — Форда используется функция searchAlgorithm, которая производит поиск кратчайшего пути из заданной вершины в графе. А с помощью функции outputWays производится вывод кратчайшего пути из заданной вершины в графе.

*3) Построение графа.*   
Построение будет состоять из двух функций. Прямое построение графа, заданного с помощью списка смежности пользователем в файле, и генерация рандомного графа с указанным пользователем количеством вершин.

Построение графа пользователем из файла осуществляется функцией inputFile. В случае нехватки данных, файла, не содержащего данных или отсутствия файла, будут брошены соответствующие исключения.

Построение случайно сгенерированного графа производится функцией inputGeneration(). Пользователю нужно будет ввести в поля меню количество вершин и количество исходящих из каждой вершины ребер. Весами ребер будут рандомные числа.  
*4) Визуализация.*

Визуализация разработана с помощью JavaFX - платформы на основе Java для создания приложений с насыщенным графическим интерфейсом.   
Файл "sample.fxml" содержит визуализацию главного меню. Файл "GUI\_prototype.fxml" содержит визуализацию дочернего окна для отрисовки итогового графа.

Вывод на экран конечного результата и пошаговое выполнение алгоритма содержится в файле GraphController.java

# 5. **План разработки и распределение ролей в бригаде**

## 5.1. **План разработки**

1) 29.06.2018 – предоставление UML-диаграммы классов программы, демонстрация демо-версии визуального интерфейса алгоритма.

2) 03.07.2018 – реализация алгоритма Беллмана – Форда, демонстрация пользовательского интерфейса.

3) 0х.07.2018 –

## 5.2. **Распределение ролей в бригаде**

В бригаде три человека, между которыми распределена работа над проектом.

|  |  |
| --- | --- |
| Кухарев М.А. | Визуализация на JavaFX, графический интерфейс. |
| Быков И. В. | Реализация алгоритма Беллмана — Форда. |
| Лавренкова Е.Л. | Тестирование программы. |

**Использованные структуры данных:**

Граф представлен списком двух вершин и весом ребра между ними.

*Класс для хранения ребер графа:*

**public class** ElementGraphWay {  
 **int from**; //начальная вершина  
 **int to**; //конечная вершина  
 **int l**; //вес ребра  
}

*Класс для хранения координат узлов графа:***public class** WaysPoint {  
 **int x**; //х-координата вершины  
 **int y**; //у-координата вершины  
 **int name**;//имя вершины  
}

# 6. **ТЕСТИРОВАНИЕ ПРОГРАММЫ**

## **6.1. Тестирование работы программы при вводе из файла in.txt**:

## 6.2. **Тестирование работы программы с помощью случайной генерации.**

# 7. **ЗАКЛЮЧЕНИЕ**

По завершению учебной практики была написана программа, выполняющая поиск кратчайших путей в графе с помощью алгоритма Беллмана — Форда для неотрицательных графов, а также была выполнена визуализация алгоритма.

Программа реализована на языке Java в интегрированной среде разработки IntelliJ IDEA, с использованием Фреймворка JavaFX.

# 8. ПРИЛОЖЕНИЕ А. ИСХОДНЫЙ КОД