«Санкт-Петербургский государственный электротехнический университет «ЛЭТИ» им. В.И.Ульянова (Ленина)»
(СПбГЭТУ «ЛЭТИ»)

ОТЧЁТ
ПО УЧЕБНОЙ ПРАКТИКЕ

Тема: РАЗРАБОТКА ВИЗУАЛИЗАТОРА АЛГОРИТМА ФЛОЙДА-УОРШАЛА

Студент гр. 4381	П.Н. Хохряков
Студент гр. 4341	В.А. Гребенюк
Студент гр. 4345	Н.Г. Пикалев
Руководитель	М.А. Фирсов

Санкт-Петербург
2026

ЗАДАНИЕ

НА УЧЕБНУЮ ПРАКТИКУ

Студент: П.Н. Хохряков Группа: 4381

Студент: В.А. Гребенюк Группа: 4341

Студент: Н.Г. Пикалев Группа: 4345

Тема практики: Разработка визуализатора алгоритма Флойда-Уоршалла на языке java

Задание на практику:

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

Алгоритм: Флойда-Уоршалла.

Сроки прохождения практики: 06.07.2026 – 18.07.2026

Дата сдачи отчета: 18.07.2026

Дата защиты отчета: 18.07.2026

Студент	П.Н. Хохряков
Студент	В.А. Гребенюк
Студент	Н.Г. Пикалев
Руководитель	М.А. Фирсов

АННОТАЦИЯ

В данной работе представлена разработка программного обеспечения для визуализации алгоритма Флойда-Уоршелла, предназначенного для поиска кратчайших путей между всеми парами вершин в ориентированном графе. Программа реализована на языке Java и предоставляет интерактивный графический интерфейс, позволяющий пошагово отслеживать работу алгоритма: видеть, какие элементы матрицы сравниваются на каждой итерации, какие вершины и рёбра задействованы, и как формируется итоговая матрица кратчайших расстояний. Поддерживаются четыре способа ввода данных: ручное редактирование, загрузка из CSV-файла, добавление/удаление вершин и генерация случайной матрицы. Программа разработана в три этапа (прототип, версия 1, версия 2) и предназначена для использования в учебных целях при изучении алгоритмов на графах.

SUMMARY

This work presents the development of software for visualizing the Floyd-Warshall algorithm, designed to find the shortest paths between all pairs of vertices in a directed graph. The program is implemented in Java and provides an interactive graphical interface that allows step-by-step tracking of the algorithm's operation: seeing which matrix elements are compared at each iteration, which vertices and edges are involved, and how the final shortest distances matrix is formed. Four input methods are supported: manual editing, loading from a CSV file, adding/removing vertices, and generating a random matrix. The program was developed in three stages (prototype, version 1, version 2) and is intended for educational purposes in studying graph algorithms.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ 5
1 ПОСТАНОВКА ЗАДАЧИ 6
1.1 Предметная область 6
1.2 Задачи практики 6
1.2.1 Выполнить вводное задание 6
1.2.2 Составить спецификацию программы и план разработки 6
1.2.3 Изучить новые методы технологии в java для выполнения командного задания 6
1.2.4 Разработать приложение в команде в соответствии со спецификацией и планом 6
2 РЕЗУЛЬТАТЫ РАБОТЫ 8
2.1. Вводное задание: 8
2.2. Составление спецификации программы и плана разработки 8
2.2.1 Исходные требования к программе 8
2.2.2 План разработки и распределение ролей в бригаде 17
2.2.3 Уточнение требований после сдачи прототипа 21
2.2.4 Уточнение требований после сдачи 1-й версии 21
2.3. Командная итеративная разработка приложения в соответствии со спецификацией и планом 23
2.3.1. Структуры данных 23
2.3.2. Основные методы 23
2.3.3. Тестирование 25
3 ИСПОЛЬЗУЕМЫЕ ТЕХНОЛОГИИ 29
4 ОТЗЫВ РУКОВОДИТЕЛЯ 31
ЗАКЛЮЧЕНИЕ 32
СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 33
ПРИЛОЖЕНИЕ А 34
ПРИЛОЖЕНИЕ Б 38
ПРИЛОЖЕНИЕ В 43

ВВЕДЕНИЕ

Предметная область данной практики охватывает теорию графов как один из фундаментальных разделов дискретной математики и информатики, а также разработку программного обеспечения для решения прикладных задач на графах.

Цель практики. Целью данной практики является закрепление и углубление теоретических знаний, полученных в ходе изучения дисциплин по алгоритмам и структурам данных, объектно-ориентированному программированию и разработке программного обеспечения, а также приобретение практических навыков командной разработки приложения.

В рамках практики ставится задача спроектировать и реализовать учебное программное средство, визуализирующее пошаговую работу алгоритма Флойда–Уоршелла, в виде приложения на языке Java с графическим интерфейсом.

1 ПОСТАНОВКА ЗАДАЧИ

1.1 Предметная область

Предметная область данной практики охватывает теорию графов как один из фундаментальных разделов дискретной математики и информатики. Одной из ключевых задач теории графов является поиск кратчайших путей между вершинами, для решения которой разработан ряд классических алгоритмов, среди которых особое место занимает алгоритм Флойда–Уоршелла, позволяющий находить кратчайшие расстояния между всеми парами вершин во взвешенном ориентированном графе за кубическое время.

Отдельным направлением, тесно связанным с алгоритмической теорией графов, является визуализация алгоритмов — область, занимающаяся наглядным представлением работы вычислительных процедур.

1.2 Задачи практики

1.2.1 Выполнить вводное задание

Ознакомиться с языком программирования java, выучить базовые методы программирования на java. Используя все полученные знания о java выполнить вводное задание.

1.2.2 Составить спецификацию программы и план разработки

Сформулировать и задокументировать требования к разрабатываемому программному продукту в виде технической спецификации, включающей описание способов ввода данных, состава и поведения элементов графического интерфейса, форматов выходных данных и перечня допустимых действий пользователя.

1.2.3 Изучить новые методы технологии в java для выполнения командного задания

Понять какие новые методы в java понадобятся для выполнения командного задания и изучить их.

1.2.4 Разработать приложение в команде в соответствии со спецификацией и планом

Спроектировать архитектуру приложения с разделением ответственности между моделью данных, графическим представлением и управляющей логикой (паттерн MVC), определить состав классов, их взаимодействие и интерфейсы обмена данными.

Реализовать модель графа и сам алгоритм Флойда–Уоршелла в виде итератора, поддерживающего пошаговое выполнение с сохранением состояния на каждой итерации циклов.

Разработать графический интерфейс, включающий холсты для отрисовки графов, таблицы матриц смежности и кратчайших расстояний, панель управления, поле описания текущего шага и панель журнала событий.

Реализовать механизмы ввода-вывода: загрузку и сохранение матриц смежности в формате CSV, ведение и экспорт журнала событий программы.

Провести тестирование разработанных компонентов, обеспечить корректную работу всех сценариев использования, описанных в спецификации.

2 РЕЗУЛЬТАТЫ РАБОТЫ

2.1. Вводное задание:

В индивидуальном задании нужно было переписать программу на язык программирования java.

Программа находит минимальное число K квадратов, из которых можно построить квадрат заданного размера N. Выводит количество, расположение и размеры всех K квадратов.

Вывод переписанной программы идентичен выводу оригинальной. (рис. 1)

```
9
6
1 1 6
1 7 3
7 1 3
4 7 3
7 4 3
7 7 3
```

Рисунок 1 – Пример выполнения программы

2.2. Составление спецификации программы и плана разработки

2.2.1 Исходные требования к программе

2.2.1.1 Требования к вводу исходных данных

Программа должна поддерживать четыре способа задания исходного графа (таблица 1). Все способы взаимозаменяемы: любой из них приводит к одному и тому же внутреннему представлению графа, после чего пользователь может запустить алгоритм.

Таблица 1 – Способы ввода

№	Способ ввода	Описание
1	Ввод через таблицу в интерфейсе	Пользователь редактирует ячейки таблицы 1 (см. 1.2.1) вручную: задаёт количество вершин, веса рёбер между парами вершин. Значение «inf» означает отсутствие ребра.

№	Способ ввода	Описание
2	Загрузка из текстового файла	Пользователь нажимает кнопку «Ввод из файла», выбирает CSV-файл с квадратной матрицей смежности; программа заполняет таблицу 1 загруженными значениями.
3	Добавление / удаление вершин кнопками	Кнопки «Добавить вершину» и «Удалить вершину» на панели управления изменяют размер матрицы: первая добавляет новую строку и столбец со значениями «inf» по умолчанию, вторая удаляет выбранную вершину (или последнюю, если выделения нет).
4	Генерация случайной матрицы смежности	Программа может сгенерировать матрицу по заданным параметрам: число вершин N, степень каждой вершины (в диапазоне от 1 до N–1), веса рёбер (случайные целые значения в заданном диапазоне).

При использовании способа 4 программа запрашивает у пользователя следующие параметры:

Число вершин N — целое число в диапазоне от 2 до 20.

Степень вершины — целое число в диапазоне от 1 до N–1; для каждой вершины программа случайно выбирает столько исходящих рёбер.

Веса рёбер — случайные целые числа в диапазоне, заданном пользователем (по умолчанию от 1 до 99).

2.2.1.2 Требования к визуализации

2.2.1.2.1 Элементы пользовательского интерфейса

Интерфейс программы состоит из семи функциональных элементов: двух холстов, двух таблиц, панели управления, поля описания текущего шага и панели журнала событий. Ниже описано назначение, внешний вид и поведение каждого элемента.

Холст 1 — вводимый граф

Холст 1 расположен в левой верхней части окна и отображает граф, заданный пользователем в таблице 1. Каждая вершина изображается кружком с номером внутри; каждое ориентированное ребро — стрелкой от исходной вершины к целевой, с подписанным весом. Вершины размещаются по кругу с равными интервалами. При любом изменении таблицы 1 (ручное редактирование, загрузка файла, добавление или удаление вершины) холст 1 автоматически перерисовывается. Пользователь может прокручивать холст колесом мыши и выделять на нём элементы.

Холст 2 — текущий граф кратчайших путей

Холст 2 расположен в правой верхней части окна и отображает текущее состояние графа кратчайших путей, соответствующее таблице 2. На нём показываются только те рёбра, вес которых на текущем шаге отличен от «inf», то есть только достижимые пары вершин. Вершины и рёбра, задействованные в текущей итерации алгоритма (то есть те, которые алгоритм в данный момент сравнивает или обновляет), подсвечиваются голубым цветом. При переходе между шагами холст 2 автоматически обновляется.

Таблица 1 — вводимая матрица смежности

Таблица 1 расположена в левой нижней части окна под холстом 1 и представляет собой редактируемую квадратную матрицу N×N. Заголовки строк и столбцов — номера вершин от 0 до N–1. В ячейке с координатами (i, j) записан вес ребра из вершины i в вершину j. Пользователь может редактировать значения ячеек напрямую с клавиатуры. Значение «inf» (или пустая ячейка) трактуется как отсутствие ребра. Диагональные ячейки (i, i) по умолчанию равны 0 и недоступны для редактирования. При изменении любого значения холст 1 немедленно перерисовывается. Ячейки, выделенные пользователем (мышью на холсте 1 или кликом по самой ячейке), подсвечиваются жёлтым цветом.

Таблица 2 — текущая матрица алгоритма

Таблица 2 расположена в правой нижней части окна под холстом 2 и отображает текущее состояние матрицы кратчайших расстояний,

вычисляемой алгоритмом. Структура таблицы 2 совпадает со структурой таблицы 1 (N×N, заголовки — номера вершин), но ячейки таблицы 2 недоступны для редактирования пользователем. На каждом шаге алгоритма ячейки, которые в данный момент сравниваются или обновляются, подсвечиваются жёлтым цветом; вершины и рёбра, соответствующие этим ячейкам, подсвечиваются голубым на холсте 2. До запуска алгоритма таблица 2 совпадает с таблицей 1.

Панель управления

Панель управления расположена в левой нижней части окна, под таблицами. Она содержит десять кнопок, сгруппированных в два ряда по пять кнопок. Каждая кнопка описана ниже (таблица 2, таблица 3, таблица 4).

Таблица 2 – Кнопки управления шагами алгоритма

Кнопка	Действие
Шаг назад	Возвращает алгоритм на один шаг назад. Если алгоритм находится на первом шаге, кнопка неактивна. Доступна начиная с версии 2.
Пуск / Пауза	Запускает автоматическое выполнение алгоритма с текущего шага. При повторном нажатии — ставит выполнение на паузу. Скорость автоматического выполнения задаётся кнопкой «Скорость kx».
Шаг вперёд	Выполняет ровно один шаг алгоритма вперёд. Если алгоритм завершён, кнопка неактивна.
Шаг N	Выполняет N шагов алгоритма вперёд за одно нажатие. Значение N вводится пользователем в поле рядом с кнопкой. Если до завершения алгоритма осталось меньше N шагов, выполняются все оставшиеся.

Таблица 3 – Кнопки ввода вывода

Кнопка	Действие
Ввод из файла	Открывает диалог выбора файла; выбранный CSV-файл загружается в таблицу 1. Если в таблице 1 есть несохранённые изменения, программа запрашивает подтверждение перезаписи.
Сохранение	Открывает диалог выбора файла; текущее содержимое таблицы 2 (матрица кратчайших расстояний) сохраняется в CSV-файл, а текущие журналы событий программы — в отдельный текстовый файл.

Таблица 4 – Кнопки управления вершинами и общие

Кнопка	Действие
Добавить вершину	Добавляет в граф новую вершину с номером N (где N — текущее количество вершин). В таблице 1 добавляется новая строка и новый столбец; все новые ячейки заполняются значением «inf», диагональная — значением 0. На холсте 1 новая вершина размещается на круговой раскладке; остальные вершины перераспределяются равномерно.
Удалить вершину	Удаляет выделенную вершину. Если ни одна вершина не выделена — удаляется последняя. Из таблиц 1 и 2 удаляются соответствующие строка и столбец, оставшиеся вершины перенумеровываются. Кнопка неактивна, если в графе меньше двух вершин.
Сброс	Возвращает алгоритм в начальное состояние: таблица 2 снова совпадает с таблицей 1, холст 2 отображает исходный граф, номер шага обнуляется. Содержимое таблицы 1 не меняется.
Скорость kx	Открывает выпадающий список для выбора множителя скорости автоматического выполнения: 0.5x, 1x, 2x, 4x. Значение по умолчанию — 1x (один шаг в секунду).

Поле описания текущего шага

Поле расположено в правой нижней части окна, рядом с панелью управления. В нём в человекочитаемом виде выводится информация о текущем шаге алгоритма.

Панель журнала событий

Панель журнала событий занимает нижнюю часть окна на всю её ширину. В ней построчно выводятся все события программы: действия пользователя, изменения состояния алгоритма, ошибки ввода-вывода. Каждая строка журнала событий содержит временную метку, тип события и текстовое описание. Содержимое панели журнала событий можно сохранить в текстовый файл кнопкой «Сохранение».

2.2.1.2.2 Схематическое представление графического интерфейса

Окно программы делится на четыре области. Верхняя половина занята двумя холстами бок о бок, средняя — двумя таблицами под ними, нижняя — панелью управления, полем описания шага и панелью журнала событий (таблица 5 и таблица 6).

Таблица 5 – Верхняя область окна программы

Холст 1	Холст 2
Таблица 1	Таблица 2
Панель управления	Поле описания шага
Панель журнала событий	

Таблица 6 – Расположение кнопок на панели управления

Шаг назад	Пуск / Пауза	Шаг вперёд	Шаг N	Добавить вершину
Сброс	Ввод из файла	Сохранение	Скорость kx	Удалить вершину

2.2.1.2.3 Взаимодействие с холстом при помощи мыши

Пользователь может взаимодействовать с холстами 1 и 2 при помощи мыши двумя способами: прокруткой и выделением элементов.

Прокрутка холста

Колесо мыши прокручивает содержимое холста по вертикали. При удержании клавиши Shift прокрутка выполняется по горизонтали. Если граф полностью помещается в холст, прокрутка не оказывает действия.

Выделение элементов на холсте

Клик левой кнопкой мыши по вершине на холсте выделяет её. Клик по ребру выделяет ребро. Выделенный элемент отображается утолщённой обводкой. Одновременно с выделением на холсте в соответствующей таблице (таблица 1 для холста 1, таблица 2 для холста 2) подсвечиваются связанные элементы:

При выделении вершины с номером i в таблице подсвечиваются строка i и столбец i — все рёбра, исходящие из этой вершины и входящие в неё.

При выделении ребра из вершины i в вершину j в таблице подсвечивается ячейка (i, j).

Обратное действие также работает: клик по ячейке (i, j) таблицы выделяет соответствующее ребро на холсте; клик по заголовку строки i или столбца j выделяет всю строку i (или весь столбец j) в таблице и вершину i (или j) вместе со всеми её исходящими или входящими рёбрами на холсте. Выделение сбрасывается кликом по пустому месту на холсте или нажатием клавиши Esc.

2.2.1.2.4 Содержимое поля описания текущего шага

Поле описания шага отображает информацию о текущем шаге алгоритма в следующем формате (таблица 7). Каждая строка поля содержит один информационный элемент. Все строки обновляются при каждом переходе между шагами.

Таблица 7 – Информация о текущем шаге алгоритма

Элемент	Описание	Пример
Номер шага	Порядковый номер шага алгоритма, начиная с 1. Шаг 1 соответствует первой итерации внешнего цикла (k = 0).	Шаг 7 из 27
Итерация циклов (k, i, j)	Текущие значения трёх индексов алгоритма Флойда-Уоршалла: k — номер промежуточной вершины внешнего цикла; i — номер исходной вершины среднего цикла; j — номер целевой вершины внутреннего цикла. Индексы принимают значения от 0 до N–1.	k = 1, i = 2, j = 3
Сравниваемые вершины и вес ребра	Номера вершин i и j, между которыми алгоритм проверяет возможность улучшения пути, и веса: текущее кратчайшее расстояние D[i][j] и альтернативный путь через вершину k — D[i][k] + D[k][j].	D[i][j] = 10, путь через k=1: D[i][k] + D[k][j] = 4 + 3 = 7
Результат сравнения	Текстовое описание того, произошло ли обновление. Если альтернативный путь короче, ячейка обновляется; иначе — остаётся без изменений.	Обновлено: D[i][j] = 7 (было 10)
Краткое описание шага	Человекочитаемая формулировка действия, выполненного на текущем шаге.	Путь из вершины 2 в вершину 3 через вершину 1 короче; матрица обновлена.

2.2.1.2.5 Подсветка элементов на шаге алгоритма

На каждом шаге алгоритма в таблице 2 и на холсте 2 подсвечиваются элементы, задействованные в текущей итерации циклов (k, i, j):

Жёлтым цветом — в таблице 2 подсвечиваются ячейки (i, j), (i, k) и (k, j). Ячейка (i, j) — та, которую алгоритм пытается улучшить; ячейки (i, k) и (k, j) — те, через которые вычисляется альтернативный путь. Если значение D[i][j]

было обновлено, ячейка остаётся жёлтой до следующего шага; если не обновлено — подсветка снимается.

Голубым цветом — на холсте 2 подсвечиваются вершины i, j, k и рёбра (i, j), (i, k), (k, j), если они существуют (то есть их вес отличен от «inf»). Вершина k — промежуточная вершина текущей итерации — выделяется особо (например, более насыщенным цветом или увеличенным размером).

2.2.1.2.6 Автоматическая синхронизация холстов и таблиц

Все четыре визуальных элемента (холст 1, холст 2, таблица 1, таблица 2) синхронизируются автоматически. Перечень случаев синхронизации:

Изменение значения в таблице 1 → холст 1 немедленно перерисовывается.

Добавление или удаление вершины → обе таблицы и холст 1 обновляются; если алгоритм был запущен, он сбрасывается.

Переход к новому шагу алгоритма → таблица 2 и холст 2 обновляются; поле описания шага заполняется новыми значениями.

Выделение элемента на холсте 1 → соответствующие ячейки таблицы 1 подсвечиваются (и наоборот).

Выделение элемента на холсте 2 → соответствующие ячейки таблицы 2 подсвечиваются (и наоборот).

2.2.1.2.7 Поведение новых вершин

При добавлении новой вершины все её рёбра по умолчанию отсутствуют: в таблице 1 новые ячейки (кроме диагональной) заполняются значением «inf». Диагональная ячейка (i, i) всегда равна 0. Пользователь может вручную задать веса рёбер для новой вершины, отредактировав соответствующие ячейки таблицы 1.

2.2.1.2.8 Отображение графа кратчайших путей на холсте 2

На холсте 2 отображаются только те рёбра, вес которых на текущем шаге отличен от «inf». Это означает, что если на каком-либо шаге алгоритма пара вершин остаётся недостижимой, ребро между ними на холсте 2 не рисуется. По мере выполнения алгоритма рёбра могут появляться (если ранее «inf»

заменяется числом) или исчезать (если ранее существовавшее ребро заменяется «inf», что в алгоритме Флойда Уоршалла не происходит, но предусмотрено для общности).

2.2.1.2.9 Полный перечень доступных действий пользователя

В таблице 8 перечислены все действия, которые пользователь может выполнить в программе (таблица 8 см. приложение А). Каждое действие может запускаться одним или несколькими способами (кнопка на панели управления, клик мыши, нажатие клавиши на клавиатуре) и имеет один результат. Этот перечень является исчерпывающим — никаких других действий программа не поддерживает.

2.2.2 План разработки и распределение ролей в бригаде

Разработка программы ведётся в три этапа: прототип, версия 1 и версия 2. Каждый этап заканчивается сдачей результата; после версии 2 следует сдача отчёта и защита. Календарный план приведён в таблице ниже (таблица 9).

Таблица 9 – Календарный план

Дата	Этап	Реализованные возможности
10.07.2026	Согласование спецификации	Требования к программе зафиксированы в настоящем документе.
13.07.2026	Сдача прототипа	Минимальный набор функций, связанных с работой алгоритма и графического интерфейса. Вывод готового результата работы алгоритма (итоговая матрица кратчайших расстояний в таблице 2 и итоговый граф кратчайших путей на холсте 2). Все элементы интерфейса присутствуют, часть кнопок неактивна.
15.07.2026	Сдача версии 1	Пошаговое выполнение алгоритма (только вперёд). Сохранение в файл и загрузка из файла. Выделение текущих элементов алгоритма на таблицах и холстах. Поле описания шага.

Дата	Этап	Реализованные возможности
17.07.2026	Сдача версии 2	Движение по алгоритму в обе стороны. Добавление шага назад. Сохранение и вывод журнала событий.
18.07.2026	Сдача отчёта	Текстовый отчёт по результатам разработки.
18.07.2026	Защита отчёта	Защита работы перед комиссией.

Примечание: реализованные возможности на каждом этапе включают, но не ограничиваются перечисленными. Возможности, добавленные на более ранних этапах, сохраняются и на более поздних.

Прототип (13.07.2026)

Прототип демонстрирует базовую работу алгоритма и графического интерфейса: пользователь вводит граф в таблицу 1 вручную, программа выполняет алгоритм Флойда Уоршалла до конца и выводит готовый результат — итоговую матрицу кратчайших расстояний в таблице 2 и итоговый граф кратчайших путей на холсте 2. Пошаговое выполнение алгоритма в прототипе не поддерживается.

Все элементы графического интерфейса присутствуют в прототипе с самого начала: оба холста, обе таблицы, все десять кнопок панели управления, поле описания шага и панель журнала событий. Однако часть элементов неактивна или не выполняет действий.

Неактивные кнопки: «Шаг вперёд», «Шаг назад», «Шаг N», «Скорость kx», «Ввод из файла», «Сохранение». Эти кнопки отображаются на панели, но нажатие на них не приводит ни к какому результату.

Активные кнопки: «Пуск / Пауза» (запускает алгоритм до завершения, без пошаговой визуализации — после нажатия таблица 2 и холст 2 сразу отображают итоговый результат), «Сброс» (очищает таблицу 2 и холст 2, возвращая их к исходному состоянию — таблица 2 снова совпадает с таблицей 1), «Добавить вершину», «Удалить вершину».

Поле описания шага в прототипе пусто (или содержит сообщение «Алгоритм не запущен» до выполнения и «Алгоритм завершён» — после).

Подсветка текущих элементов алгоритма отсутствует, так как пошаговое выполнение не реализовано. Панель журнала событий работает и фиксирует запуск программы, запуск алгоритма, добавление/удаление вершин и завершение алгоритма.

Версия 1 (15.07.2026) — подробное описание выполнения алгоритма

Версия 1 добавляет к прототипу пошаговое выполнение алгоритма, загрузку и сохранение файлов, подсветку текущих элементов алгоритма и заполнение поля описания шага. Выполнение алгоритма в версии 1 выглядит следующим образом.

Исходное состояние. Пользователь задаёт граф одним из двух способов: редактирует таблицу 1 вручную (как в прототипе) или нажимает кнопку «Ввод из файла» и выбирает CSV-файл с матрицей смежности. После загрузки файла таблица 1 заполняется значениями, холст 1 перерисовывается. Таблица 2 на этом этапе совпадает с таблицей 1 (алгоритм ещё не запущен); холст 2 отображает тот же граф, что и холст 1. Поле описания шага пусто или содержит сообщение «Алгоритм не запущен». Программа находится в состоянии «Ожидание ввода».

Выполнение одного шага. Пользователь нажимает кнопку «Шаг вперёд». Программа переходит в состояние «Алгоритм на паузе» и выполняет одну итерацию циклов (k, i, j) алгоритма Флойда Уоршалла. Конкретно: программа берёт текущие значения индексов k, i, j (начиная с k=0, i=0, j=0), сравнивает значение D[i][j] с суммой D[i][k] + D[k][j]; если сумма меньше — ячейка D[i][j] обновляется. После этого:

В таблице 2 ячейки (i, j), (i, k), (k, j) подсвечиваются жёлтым.

На холсте 2 вершины i, j, k и рёбра между ними подсвечиваются голубым.

Поле описания шага заполняется: номер шага, значения k, i, j, сравниваемые веса, результат сравнения, краткое текстовое описание (см. раздел 1.2.4).

В панель журнала событий добавляется запись типа STATE с информацией о шаге.

Автоматическое выполнение. Пользователь может нажать кнопку «Пуск / Пауза» — алгоритм начнёт выполняться автоматически со скоростью, заданной множителем kx (по умолчанию 1x — один шаг в секунду). На каждом шаге выполняются те же действия, что и при одиночном шаге. Повторное нажатие «Пуск / Пауза» останавливает выполнение; текущий шаг сохраняется. Кнопка «Шаг N» выполняет N шагов подряд без остановок (но без анимации — результат отображается только после завершения всех N шагов).

Завершение алгоритма. Когда индексы k, i, j доходят до последних значений (k = N–1, i = N–1, j = N–1), алгоритм считается завершённым. Программа переходит в состояние «Алгоритм завершён». Таблица 2 содержит итоговую матрицу кратчайших расстояний; холст 2 отображает итоговый граф кратчайших путей (только рёбра с весом ≠ inf). Кнопки «Шаг вперёд», «Пуск / Пауза», «Шаг N» становятся неактивными; кнопка «Шаг назад» остаётся неактивной до версии 2. Пользователь может нажать «Сброс» (алгоритм возвращается в исходное состояние) или «Сохранение» (открывается диалог выбора файла; итоговая матрица сохраняется в CSV, журнал событий — в текстовый файл).

Что НЕ входит в версию 1. Кнопка «Шаг назад» присутствует на панели, но остаётся неактивной; движение назад по шагам появится в версии 2. До этого момента вернуться к предыдущему состоянию можно только кнопкой «Сброс» (которая откатывает алгоритм к началу, а не на один шаг). Сохранение журнала событий в файл появится в версии 2; в версии 1 журнал событий отображаются в панели, но не сохраняются.

Версия 2 (17.07.2026)

Версия 2 добавляет движение по алгоритму в обе стороны: становится доступной кнопка «Шаг назад». Программа хранит историю всех состояний таблицы 2 и при нажатии «Шаг назад» восстанавливает предыдущее состояние. Также в версии 2 добавляется сохранение журнала событий в текстовый файл вместе с сохранением матрицы. Из распределения ролей должно быть понятно, какие части приложения были разработаны каждым студентом.

2.2.2.2 Распределение ролей в бригаде

Ниже перечислены члены бригады и их роли. Для каждого участника указаны конкретные задачи, за которые он отвечает.

Гребенюк В.А. — проектировщик и разработчик интерфейса

Общий дизайн графического интерфейса: расположение элементов, цветовая схема, шрифты.

Реализация графического интерфейса: холсты, таблицы, панель управления, поле описания шага, панель журнала событий.

Пикалев Н.Г. — разработчик алгоритма и генератора данных

Реализация алгоритма Флойда Уоршалла с пошаговым выполнением.

Генератор входных данных: случайная генерация матрицы смежности по заданным параметрам.

Хохряков П.Н. — разработчик ввода-вывода, логирования и тестирования

Реализовать загрузку и сохранение CSV-файлов (см. разделы 1.1.1 и 1.3.1).

Реализовать систему логирования состояний и событий программы (см. раздел 1.3.2).

Добавить валидацию входных данных: проверка квадратности матрицы, корректности весов, диапазона вершин.

Написать unit-тесты на модуль загрузки и сохранения CSV.

Интеграционное и сквозное тестирование: проверка связки интерфейса, алгоритма и ввода-вывода.

2.2.3 Уточнение требований после сдачи прототипа

Прототип: "пользователь вводит граф в таблицу 1 вручную, программа выполняет алгоритм Флойда-Уоршалла до конца и выводит готовый результат". Однако редактирование таблицы 1 игнорируется при выполнении алгоритма.

2.2.4 Уточнение требований после сдачи 1-й версии

- Увеличить количество смысловых цветов в подсветке:

- отдельные цвета для вершин k, i, j (для k-вершины это и так было предусмотрено);
- при обновлении в матрице ячейку и соответствующее ребро выделять зелёным.

- В поле рядом с "Шаг N" можно ввести не только число, но и букву k, i, или j:
  - если введено "j", то алгоритм выполняет от 1 до N шагов, переходя к началу следующей средней итерации (0 шагов в конечном состоянии);
  - если введено "i", то алгоритм выполняет от 1 до N*N шагов, переходя к началу следующей крупной итерации (0 шагов в конечном состоянии);
  - если введено "k", то алгоритм выполняет все оставшиеся до конца шаги.
- Добавить переключатель "фиксировать вершины" (по умолчанию неактивный), при включении которого движение вершин становится невозможным.
- Снимать подсветку не только по клику по пустому месту на холсте, но и по пустому месту в области матрицы.
- В панели логов выводить сообщение выполнения N шагов, а именно:
  - "Шаг N: выполнение ... шаг(ов)" (... — указанное в поле значение);
  - далее запись для каждого шага;
  - далее "Выполнено ... шаг(ов)" (... — сколько реально шагов было выполнено).
- Высоту поля описания текущего шага чуть увеличить, чтобы текст влезал, когда происходит обновление.

2.3. Командная итеративная разработка приложения в соответствии со спецификацией и планом

2.3.1. Структуры данных

2.3.1.1 Модели данных

Матрица смежности – тип Integer[][]. Хранит веса рёбер графа. null означает отсутствие ребра (inf), 0 на диагонали – расстояние от вершины до себя. Используется в Graph, FloydWarshallExecutor.

Graph — тип класс. Инкапсулирует граф: размер, матрицу, операции добавления/удаления вершин. Гарантирует инварианты (диагональ = 0).

StepResult – тип record. Хранит результат одного шага алгоритма: индексы (k, i, j), флаг обновления, описание, текущую матрицу. Используется в FloydWarshallExecutor, Controller.

FloydWarshallExecutor – тип класс. Итератор алгоритма. Хранит текущее состояние циклов (k, i, j) и рабочую матрицу dist.

2.3.1.2 Перечисления

State. Значения WAITING_INPUT, ALGORITHM_FINISHED, PAUSED, RUNNING. Состояния программы (по спецификации – 5 состояний, включая ошибку I/O)

ButtonId. Значения START_PAUSE, RESET, ADD_VERTEX, REMOVE_VERTEX, STEP_FORWARD, STEP_BACK, STEP_N, SPEED, LOAD_FILE, SAVE. Идентификаторы кнопок панели управления.

SelectionType. Значения VERTEX, EDGE, NONE. Тип выделенного элемента на холсте.

Logger.Type. Значения INFO, ACTION, STATE, ERROR. Типы событий для логирования.

2.3.1.3 Вспомогательные структуры

LogEntry – Запись журнала событий: временная метка + тип + сообщение.

CSV-данные – Временная структура для записи/сериализации матрицы в файл.

2.3.2. Основные методы

2.3.2.1 Класс Graph – работа с графом

Graph(int n) – Создаёт пустой граф на n вершин (диагональ = 0, остальное = null)

Graph(Graph other) – Конструктор копирования (для безопасной передачи в алгоритм)

size() – Возвращает количество вершин

get(i, j) – Возвращает вес ребра (или null)

set(i, j, v) – Устанавливает вес ребра (игнорирует диагональ)

hasEdge(i, j) – Проверяет существование ребра

snapshot() – Возвращает глубокую копию матрицы

replaceMatrix(newMatrix) – Полностью заменяет матрицу с валидацией

addVertex() – Добавляет вершину (с проверкой лимита 20)

removeVertex(index) – Удаляет вершину с перенумерацией (с проверкой минимума 2)

2.3.2.2 Класс FloydWarshallExecutor – итератор алгоритма

FloydWarshallExecutor(source) – Инициализирует исполнитель с копией исходной матрицы

stepForward() – Выполняет один шаг (k, i, j), возвращает StepResult

isFinished() – Проверяет, завершён ли алгоритм

totalSteps(n) – Возвращает общее количество шагов (n^3)

2.3.2.3 Класс MatrixOps – утилиты для матриц

deepCopy(source) – Создаёт глубокую копию матрицы через System.arraycopy

2.3.2.4 Класс Controller – связывание модели и представления

wire() – Подключает все обработчики событий между View и моделью

wireSelectionSync(...) – Настраивает двустороннюю синхронизацию выделения между холстом и таблицей

onButton(ButtonId) – Диспетчер нажатий кнопок

doStartPause() – Запуск/пауза алгоритма

doStepForward() – Один шаг вперёд (использует FloydWarshallExecutor)

doStepBack() – Один шаг назад (восстанавливает из истории – версия 2)

doReset() – Сброс алгоритма к начальному состоянию

doAddVertex() / doRemoveVertex() – Управление вершинами

applyCellEdit(i, j, value) – Обработка редактирования ячейки таблицы 1 с валидацией

rebuildAll() – Обновляет обе таблицы

2.3.2.5 Методы представления (View)

GraphCanvas – setGraph(), selectVertex(), selectEdge(), clearSelection(), setSelectionListener(), paintComponent() (отрисовка)

MatrixTableView – rebuild(), toMatrix(), setEditListener(), setSelectionListener(), highlightCell(), clearSelectionSync()

ControlPanel – setListener(), setButtonEnabled() (активация/деактивация кнопок по состоянию)

Logger – log(type, message), saveToFile()

2.3.3. Тестирование

2.3.3.1. Тестирование графического интерфейса

Результаты тестирования графического интерфейса с работой алгоритма представлены на рисунках (рис. 2, рис. 3, рис. 4, рис. 5, рис. 6, рис. 7, рис. 8, рис. 9)

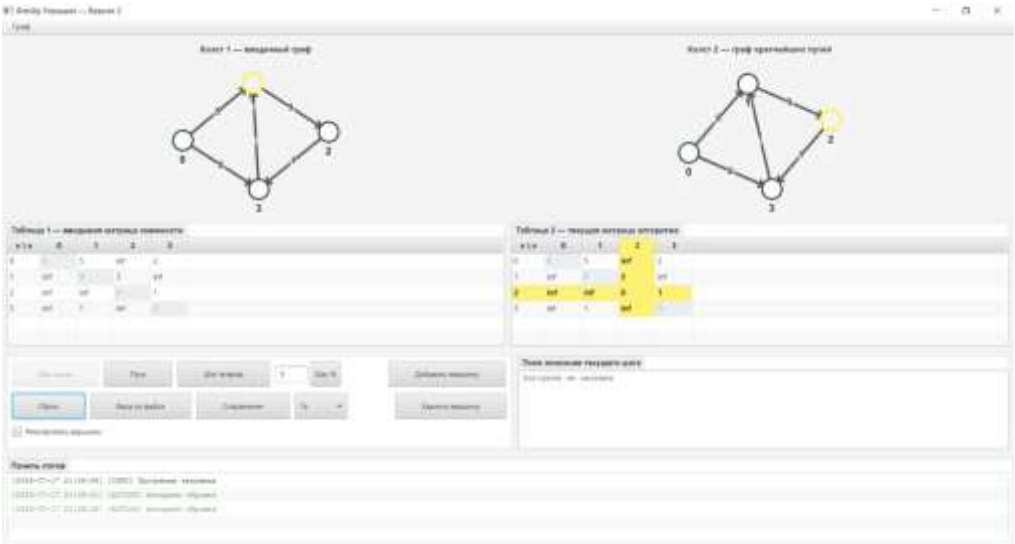

Рисунок 2 — Выделение вершины.

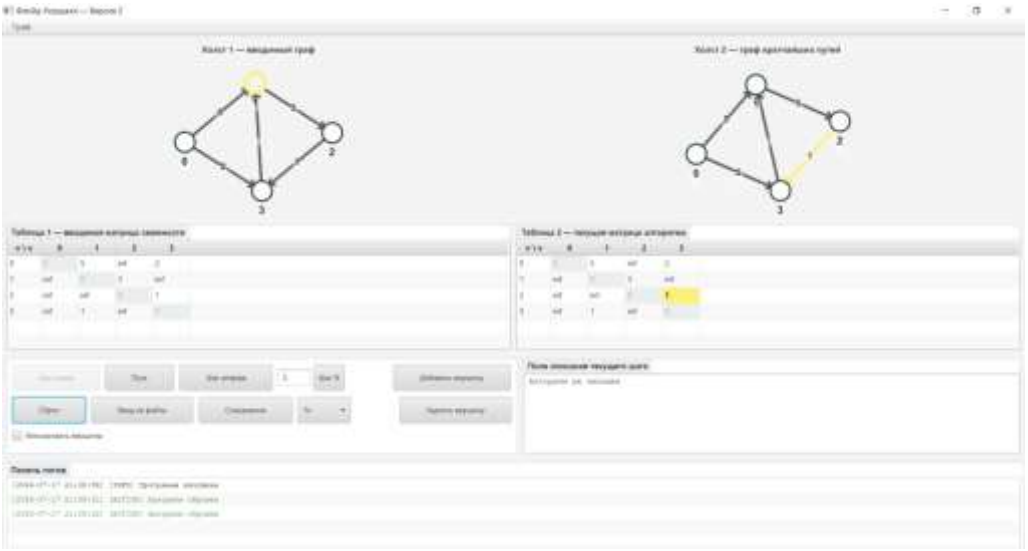

Рисунок 3 — Выделение ребра.

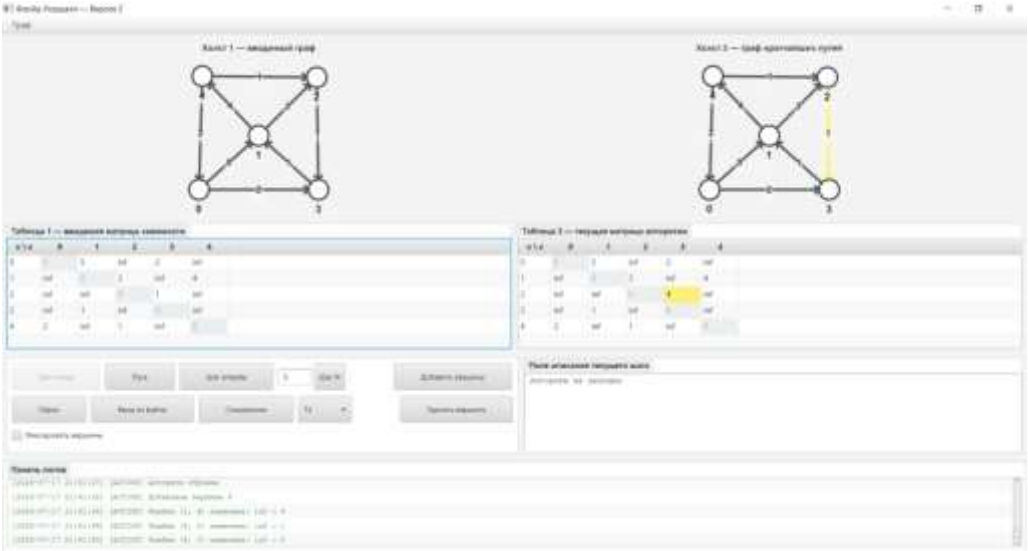

Рисунок 4 — Добавление ребра.

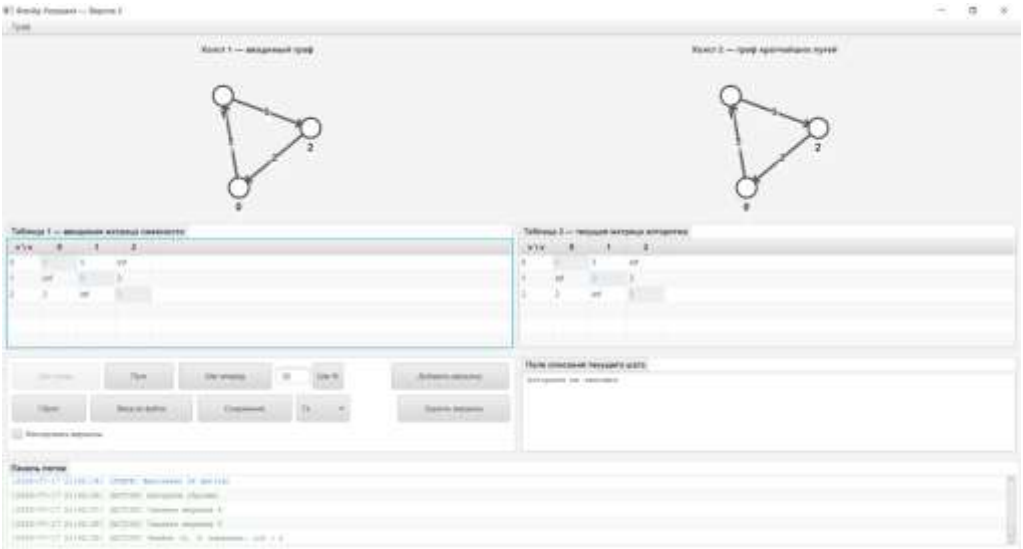

Рисунок 5 — Удаление ребра.

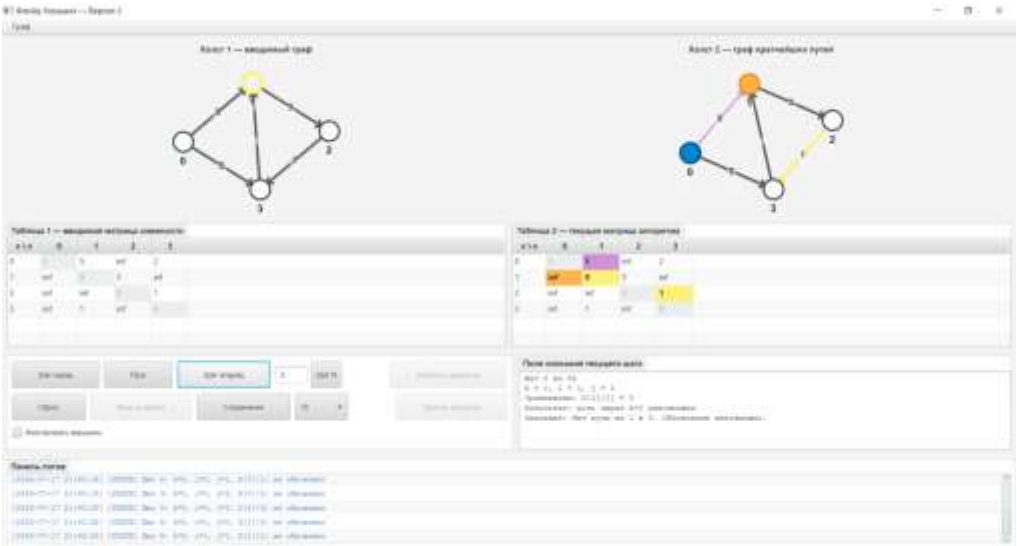

Рисунок 6 — Работа алгоритма Флойда-Уоршалла.

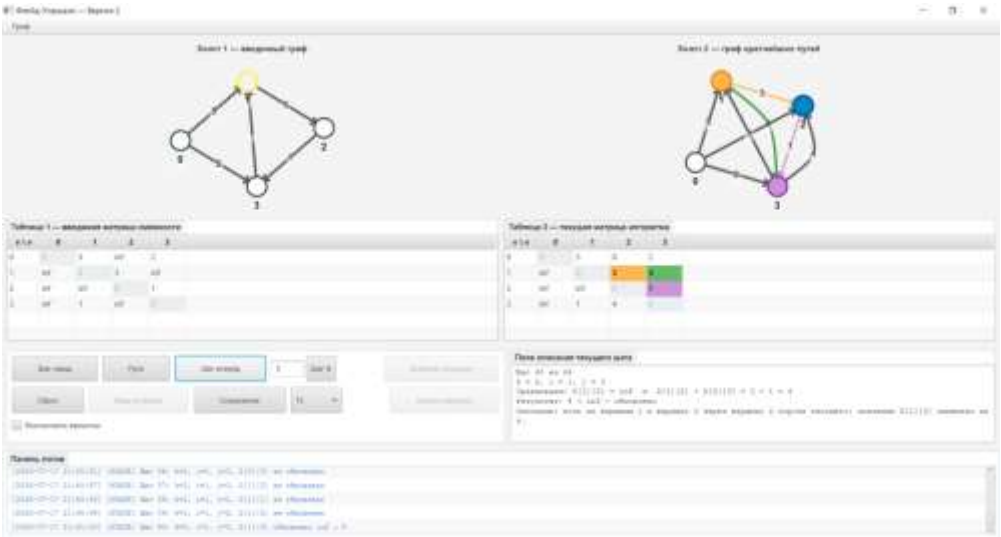

Рисунок 7 — Работа алгоритма Флойда-Уоршалла.

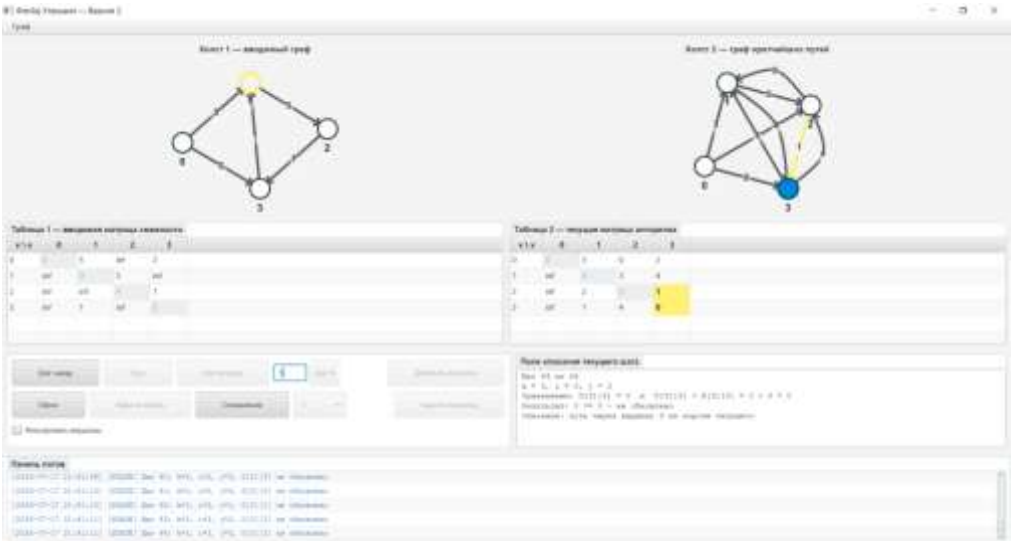

Рисунок 8 — Завершение алгоритма.

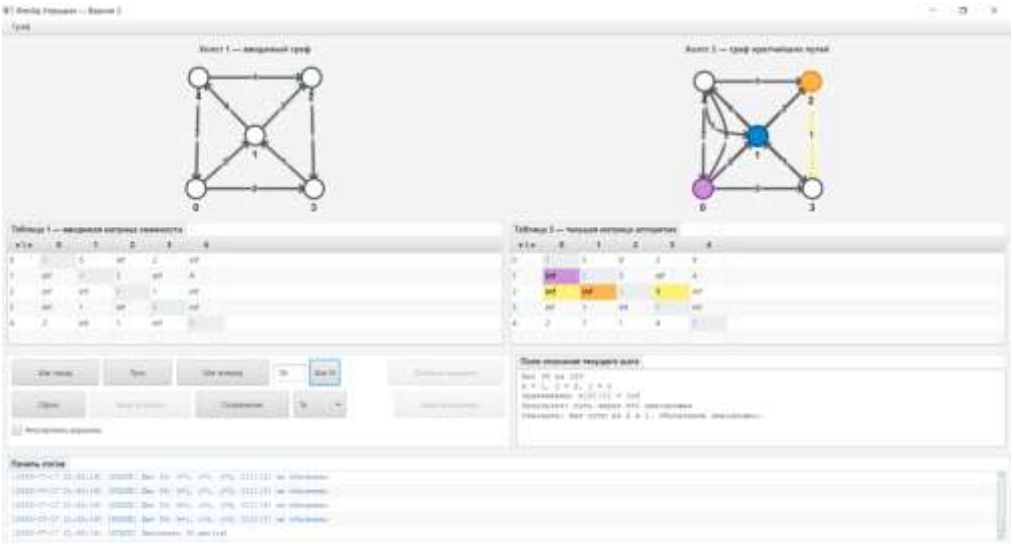

Рисунок 9 — Прохождение на N шагов вперёд.

2.3.3.2. Тестирование алгоритма

Список тестов приведен в таблице 10 (Приложение Б)

В список существующих сообщений состояний программы приведен в таблице 11 (Приложение В).

3 ИСПОЛЬЗУЕМЫЕ ТЕХНОЛОГИИ

Использованный язык программирования: Java (21)

Java (21)

Современная версия языка программирования Java с поддержкой актуальных возможностей: records (неизменяемые классы данных), pattern matching, enhanced switch, text blocks. Обеспечивает строгую типизацию, кроссплатформенность и высокую производительность для реализации алгоритмов и бизнес-логики приложения.

Использованные библиотеки:

JavaFX (0.1.0)

Современная библиотека для создания графических интерфейсов (GUI) в Java. Пришла на смену Swing, предлагая более гибкую архитектуру, поддержку CSS-стилизации, FXML для описания интерфейса, анимации и привязки данных (bindings). Используется для создания основного окна программы, таблиц, кнопок и панелей.

JavaFXSmartGraph (2.3.0)

Специализированная библиотека для интерактивного отображения графов в JavaFX. Предоставляет готовые компоненты для визуализации вершин и рёбер, поддерживает автоматическую раскладку графа (force-directed layout), перетаскивание вершин мышью, выделение элементов и анимацию. Значительно упрощает реализацию холстов 1 и 2 по сравнению с ручной отрисовкой через Graphics2D.

JUnit

Стандартный фреймворк для модульного тестирования (unit testing) в Java. Позволяет писать автоматизированные тесты для проверки корректности работы отдельных компонентов программы: алгоритма Флойда-Уоршала, загрузки/сохранения CSV-файлов, валидации входных данных. Обеспечивает надёжность кода и упрощает поиск ошибок при изменениях.

Gradle (9.6.1)

Современная система автоматической сборки (build tool) для Java-проектов. Управляет зависимостями (автоматически скачивает JavaFX, SmartGraph, JUnit из репозиториев), компилирует код, запускает тесты, создаёт исполняемый JAR-файл. Использует декларативный синтаксис в файле build.gradle для описания конфигурации проекта.

4 ОТЗЫВ РУКОВОДИТЕЛЯ

ЗАКЛЮЧЕНИЕ

В ходе прохождения практики была выполнена разработка программного средства для визуализации алгоритма Флойда–Уоршелла, предназначенного для поиска кратчайших путей между всеми парами вершин в ориентированном взвешенном графе. Работа велась в соответствии с утверждённой спецификацией и календарным планом, предусматривающим три последовательных этапа: прототип (13.07.2026), версия 1 (15.07.2026) и версия 2 (17.07.2026). Все этапы были пройдены в установленные сроки.

В результате прохождения практики был разработан полнофункциональный программный продукт, реализующий все требования спецификации. Программа поддерживает четыре способа ввода данных (редактирование таблицы, загрузка из CSV, добавление/удаление вершин, генерация случайной матрицы), пошаговое выполнение алгоритма Флойда–Уоршелла с возможностью движения вперёд и назад, автоматическое выполнение с регулируемой скоростью, синхронизированную визуализацию на холстах и в таблицах с подсветкой текущих элементов, ведение журнала событий и сохранение результатов в файлы. Приложение прошло все три запланированных этапа разработки и готово к демонстрации и защите

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Репозиторий с кодом проекта [Электронный ресурс]. URL: https://github.com/GemGameRU/practice_2026

ПРИЛОЖЕНИЕ А

Таблица 8 – Действия, которые пользователь может выполнить в программе

№	Действие	Триггер	Результат
1	Шаг вперёд	Кнопка «Шаг вперёд» или клавиша → (стрелка вправо) на клавиатуре	Алгоритм выполняет одну итерацию циклов (k, i, j) вперёд. Таблица 2, холст 2 и поле описания шага обновляются.
2	Шаг назад	Кнопка «Шаг назад» или клавиша ← (стрелка влево) на клавиатуре	Алгоритм возвращается на одну итерацию назад. Состояние таблицы 2 и холста 2 восстанавливается из истории. Доступно с версии 2.
3	Запуск автоматического выполнения	Кнопка «Пуск / Пауза» (когда алгоритм на паузе)	Алгоритм начинает выполняться автоматически с текущего шага со скоростью, заданной множителем kx. Кнопка меняет надпись на «Пауза».
4	Пауза автоматического выполнения	Кнопка «Пуск / Пауза» (когда алгоритм выполняется)	Автоматическое выполнение останавливается. Кнопка меняет надпись на «Пуск». Текущий шаг сохраняется.
5	Выполнить N шагов	Кнопка «Шаг N» (после ввода N в соседнее поле)	Алгоритм выполняет ровно N шагов вперёд. Если до завершения осталось меньше N шагов — выполняются все оставшиеся, после чего выводится сообщение о завершении.
6	Изменить скорость	Выпадающий список «Скорость kx»	Устанавливает новый множитель скорости автоматического выполнения (0.5x, 1x, 2x или 4x). Применяется немедленно.

№	Действие	Триггер	Результат
7	Сбросить алгоритм	Кнопка «Сброс»	Алгоритм возвращается в начальное состояние (шаг 0). Таблица 2 снова совпадает с таблицей 1, холст 2 отображает исходный граф. Содержимое таблицы 1 не меняется. История шагов очищается.
8	Добавить вершину	Кнопка «Добавить вершину»	В граф добавляется новая вершина с номером N. В таблицах 1 и 2 добавляются строка и столбец со значениями «inf» (диагональ — 0). Холст 1 перерисовывается. Если алгоритм был запущен — он сбрасывается.
9	Удалить вершину	Кнопка «Удалить вершину»	Удаляется выделенная вершина (или последняя, если выделения нет). Из таблиц 1 и 2 удаляются соответствующие строка и столбец. Оставшиеся вершины перенумеровываются. Холсты перерисовываются. Алгоритм сбрасывается.
10	Загрузить матрицу из файла	Кнопка «Ввод из файла»	Открывается диалог выбора файла. После выбора CSV-файла его содержимое загружается в таблицу 1. Холст 1 перерисовывается. Алгоритм сбрасывается. В панель журнала событий добавляется запись о загрузке.

№	Действие	Триггер	Результат
11	Сохранить матрицу и журнал событий	Кнопка «Сохранение»	Открывается диалог выбора файла. Текущая матрица из таблицы 2 сохраняется в выбранный CSV-файл; журнал событий программы сохраняются в отдельный текстовый файл с тем же именем и расширением .txt.
12	Редактировать ячейку таблицы 1	Двойной клик по ячейке таблицы 1 + ввод значения с клавиатуры	Значение ячейки обновляется. Холст 1 немедленно перерисовывается. Если алгоритм был запущен — он сбрасывается. В панель журнала событий добавляется запись об изменении.
13	Прокрутить холст	Колесо мыши над холстом (Shift + колесо для горизонтальной прокрутки)	Содержимое холста прокручивается. Если граф полностью помещается — действие не оказывает эффекта.
14	Выделить вершину на холсте	Клик левой кнопкой мыши по вершине на холсте 1 или холсте 2	Вершина выделяется утолщённой обводкой. В соответствующей таблице подсвечиваются строка и столбец с тем же номером.
15	Выделить ребро на холсте	Клик левой кнопкой мыши по ребру на холсте 1 или холсте 2	Ребро выделяется. В соответствующей таблице подсвечивается ячейка (i, j) для ребра из вершины i в вершину j.
16	Выделить ячейку в таблице	Клик по ячейке (i, j) таблицы 1 или таблицы 2	Ячейка подсвечивается. На соответствующем холсте выделяется ребро (i, j) (если оно существует).

№	Действие	Триггер	Результат
17	Выделить строку/столбец в таблице	Клик по заголовку строки i или столбца j в таблице 1 или таблице 2	Подсвечивается вся строка i (или весь столбец j) таблицы. На соответствующем холсте выделяется вершина i (или j) и все рёбра, исходящие из этой вершины (для строки) или входящие в неё (для столбца).
18	Снять выделение	Клик левой кнопкой мыши по пустому месту на холсте или клавиша Esc на клавиатуре	Все выделения на холстах и в таблицах снимаются.

ПРИЛОЖЕНИЕ Б

Таблица 10 – Тесты, которые успешно проход программа

№	Название теста	Класс	Тип	Ожидаемый результат
1	Создание графа с минимальным размером (2)	GraphTest	Unit	Граф создан, диагональ = 0, рёбра = null
2	Создание графа с максимальным размером (20)	GraphTest	Unit	Граф создан успешно
3	Ошибка при размере меньше 2	GraphTest	Unit	Выбрасывается IllegalArgumentException
4	Ошибка при размере больше 20	GraphTest	Unit	Выбрасывается IllegalArgumentException
5	Установка положительного веса ребра	GraphTest	Unit	Вес сохранён корректно
6	Установка null (бесконечность)	GraphTest	Unit	Ребро удалено (null)
7	Ошибка при отрицательном весе	GraphTest	Unit	Выбрасывается IllegalArgumentException
8	Ошибка при нулевом весе	GraphTest	Unit	Выбрасывается IllegalArgumentException
9	Диагональ защищена от редактирования	GraphTest	Unit	Значение осталось 0
10	Добавление вершины	GraphTest	Unit	Размер увеличился, новая вершина инициализирована
11	Ошибка при добавлении вершины сверх максимума	GraphTest	Unit	Выбрасывается IllegalStateException
12	Удаление вершины	GraphTest	Unit	Размер уменьшился

№	Название теста	Класс	Тип	Ожидаемый результат
13	Ошибка при удалении из графа с 2 вершинами	GraphTest	Unit	Выбрасывается IllegalStateException
14	Ошибка при некорректном индексе вершины	GraphTest	Unit	Выбрасывается IndexOutOfBoundsExceptio n
15	hasEdge корректно определяет наличие ребра	GraphTest	Unit	Возвращает true/false корректно
16	replaceMatrix заменяет матрицу	GraphTest	Unit	Матрица заменена, размер обновлён
17	replaceMatrix ошибка при неквадратной матрице	GraphTest	Unit	Выбрасывается IllegalArgumentException
18	Глубокое копирование создаёт независимую копию	MatrixOpsTest	Unit	Оригинал не изменился
19	Глубокое копирование сохраняет null значения	MatrixOpsTest	Unit	null значения сохранены
20	Глубокое копирование матрицы 1x1	MatrixOpsTest	Unit	Копия идентична оригиналу
21	Простой граф с известным результатом	FloydWarshallTest	Unit	Кратчайшие пути вычислены верно
22	Граф с недостижимыми вершинами	FloydWarshallTest	Unit	Недостижимые вершины остались null
23	Граф только с диагональю (изолированные вершины)	FloydWarshallTest	Unit	Только диагональ = 0, остальное null
24	Полный граф	FloydWarshallTest	Unit	Все кратчайшие пути найдены

№	Название теста	Класс	Тип	Ожидаемый результат
25	Граф с циклом	FloydWarshallTest	Unit	Кратчайшие пути через цикл вычислены
26	totalSteps вычисляет N^3	FloydWarshallTest	Unit	Возвращает N^3 корректно
27	Приемлемая матрица 3x3 с inf и пробелами	CsvLogicTest	Unit	Матрица загружена корректно
28	Регистронезависимый INF/Inf/inf	CsvLogicTest	Unit	Все inf распознаны как null
29	Диагональ принудительно обнуляется	CsvLogicTest	Unit	Диагональ автоматически = 0
30	Минимальный размер 2x2	CsvLogicTest	Unit	Загружена успешно
31	Пустые строки игнорируются	CsvLogicTest	Unit	Пустые строки пропущены
32	Ошибка: неквадратная матрица	CsvLogicTest	Unit	Выбрасывается IllegalArgumentException
33	Ошибка: матрица < 2 вершин	CsvLogicTest	Unit	Выбрасывается IllegalArgumentException
34	Ошибка: матрица > 20 вершин	CsvLogicTest	Unit	Выбрасывается IllegalArgumentException
35	Ошибка: отрицательный вес	CsvLogicTest	Unit	Выбрасывается IllegalArgumentException
36	Ошибка: нулевой вес вне диагонали	CsvLogicTest	Unit	Выбрасывается IllegalArgumentException
37	Ошибка: нечисловое значение	CsvLogicTest	Unit	Выбрасывается IllegalArgumentException
38	Ошибка: пустой файл	CsvLogicTest	Unit	Выбрасывается IllegalArgumentException
39	Сохранение: null записывается как ‘inf’	CsvLogicTest	Unit	null записаны как “inf”

№	Название теста	Класс	Тип	Ожидаемый результат
40	Round-trip: сложная матрица 4x4	CsvLogicTest	Unit	Данные идентичны после цикла
41	Сохранение сообщений	CsvLogicTest	Unit	Файл создан, данные записаны
42	Связка Graph → Executor: корректная передача матрицы	FloydWarshallIntegrationTest	Интеграционный	Матрица скопирована, изменение Executor не влияет на Graph
43	Сквозной: Graph → Executor → полный прогон → корректный результат	FloydWarshallIntegrationTest	Сквозной	Кратчайшие пути вычислены верно
44	Сквозной: Пошаговый Executor идентичен полному FloydWarshall.run	FloydWarshallIntegrationTest	Сквозной	Результаты идентичны
45	Сквозной: откат и повтор дают идентичное состояние	FloydWarshallIntegrationTest	Сквозной	Результат совпадает с первым прогоном
46	Интеграционный: replaceMatrix в Graph → новый Executor	FloydWarshallIntegrationTest	Интеграционный	Новый результат отражает изменения
47	Интеграционный: граф без рёбер — ноль обновлений	FloydWarshallIntegrationTest	Интеграционный	updateCount = 0
48	Сквозной: режимы j/i/k корректно вычисляют количество шагов	FloydWarshallIntegrationTest	Сквозной	targetSteps вычислен верно
49	MatrixOps.deepCopy создаёт независимый снимок для истории	FloydWarshallIntegrationTest	Интеграционный	Изменение оригинала не влияет на копию

№	Название теста	Класс	Тип	Ожидаемый результат
50	Снимки в истории сохраняют состояние на каждом шаге	FloydWarshallIntegrationTest	Интеграционный	Снимки независимы и корректны

ПРИЛОЖЕНИЕ В

Таблица 11 – существующие сообщения, отображающие состояния программы

Шаблон сообщения	Тип	Где формируется	Триггер
Программа запущена	INFO	App.start()	Запуск JavaFX-приложения
Диагональный элемент [{i}][{j}] автоматически изменён на 0	INFO	Controller.loadCsvMatrix()	Загрузка CSV, где на диагонали стоит не 0
Фиксация вершин: вкл/выкл	ACTION	Controller.wire()	Переключение CheckBox «Фиксировать вершины»
Скорость: {N}x	ACTION	Controller.onSpeedChanged()	Выбор значения в ComboBox скорости
Алгоритм сброшен	ACTION	Controller.doReset()	Нажатие кнопки «Сброс»
Добавлена вершина {ID}	ACTION	Controller.doAddVertex()	Нажатие кнопки «Добавить вершину»
Удалена вершина {ID}	ACTION	Controller.doRemoveVertex()	Нажатие кнопки «Удалить вершину»
Ячейка ({i}, {j}) изменена: {old} → {new}	ACTION	Controller.applyCellEdit()	Редактирование ячейки в Таблице 1
Загружен файл: {имя} ({N} вершин)	ACTION	Controller.doLoadFile()	Успешная загрузка и валидация CSV
Сохранено: {файл_матрицы}, {файл_лога}	ACTION	Controller.doSaveFile()	Успешное сохранение на диск
Сгенерирован случайный граф: N={N}, степень={D}	ACTION	Controller.generateRandomGraph()	Успешная генерация через диалог
Алгоритм запущен	STATE	Controller.ensureAlgorithmStarted()	Первый шаг (переход WAITING → PAUSED)

Шаблон сообщения	Тип	Где формируется	Триггер
Шаг {N} (из истории): k={k}, i={i}, j={j}	STATE	Controller.doStepForward()	Шаг вперёд из кэша (откат + повтор)
Шаг {N}: k={k}, i={i}, j={j}, D[{i}][{j}] обновлено/не обновлено	STATE	Controller.doStepForward()	Выполнение нового шага алгоритма
Алгоритм завершён (всего шагов: {N})	STATE	Controller.doStepForward()	Достижение k=N (последний шаг)
Шаг назад: возврат к началу	STATE	Controller.doStepBackward()	Откат к шагу 0
Шаг назад: откат к шагу {N}	STATE	Controller.doStepBackward()	Откат на один шаг из истории
Шаг N: выполнение {N} шаг(ов)	STATE	Controller.doStepN()	Нажатие «Шаг N» с числом
Шаг N (режим {j/i/k}): выполнение {N} шаг(ов)	STATE	Controller.doStepByIteration()	Нажатие «Шаг N» с буквой j, i или k
Выполнено {N} шаг(ов)	STATE	Controller.doStepN() / doStepByIteration()	Итог пакетного выполнения
Шаг N (режим {j/i/k}): 0 шагов	STATE	Controller.doStepByIteration()	Итерация уже завершена
Шаг N (режим {j/i/k}): 0 шагов (алгоритм завершён)	STATE	Controller.doStepByIteration()	Алгоритм полностью finished
Автоматическое выполнение возобновлено	STATE	Controller.doStartPause()	Нажатие «Пуск»
Пауза на шаге {N}	STATE	Controller.doStartPause()	Нажатие «Пауза»

Шаблон сообщения	Тип	Где формируется	Триггер
Вес ребра должен быть положительным (ячейка {i}, {j})	ERROR	Controller.applyCellEdit()	Ввод ≤ 0 в ячейку Таблицы 1
Некорректное значение ячейки ({i}, {j}): {val}	ERROR	Controller.applyCellEdit()	Ввод нечислового значения в ячейку
N должно быть числом или буквой k, i, j	ERROR	Controller.doStepN()	Ввод посторонних символов в поле «Шаг N»
N должно быть положительным числом	ERROR	Controller.doStepN()	Ввод 0 или отрицательного числа
Число вершин должно быть от 2 до 20	ERROR	Controller.generateRandomGraph()	Неверный N в диалоге генерации
Степень вершины должна быть от 1 до N-1	ERROR	Controller.generateRandomGraph()	Неверная степень в диалоге
Мин. вес не может быть больше макс. веса	ERROR	Controller.generateRandomGraph()	minW > maxW в диалоге
Достигнуто максимальное число вершин (20)	ERROR	Controller.doAddVertex() (из Graph)	Попытка добавить 21-ю вершину
В графе должно быть не менее двух вершин	ERROR	Controller.doRemoveVertex() (из Graph)	Попытка удалить при n=2
Некорректное значение скорости: {val}	ERROR	Controller.onSpeedChanged()	Внутренняя ошибка полученной скорости
Ошибка валидации файла: {сообщение}	ERROR	Controller.doLoadFile()	CSV не прошёл валидацию (6 подтипов*)
Не удалось загрузить файл: {сообщение}	ERROR	Controller.doLoadFile()	IO-ошибка чтения файла
Не удалось сохранить файл: {сообщение}	ERROR	Controller.doSaveFile()	IO-ошибка записи файла