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

ОТЧЕТ

по учебной практике

Тема: Топологическая сортировка

Студент гр. 8304 _____ Бутко А.М

Студент гр. 8304 _____ Нам Ё Себ

Студент гр. 8304 _____ Самакаев Д.И.

Руководитель _____ Размочаева Н.В.

Санкт-Петербург
2020

ЗАДАНИЕ
НА УЧЕБНУЮ ПРАКТИКУ

Студент Бутко А.М. группы 8304

Студент Нам Ё Себ группы 8304

Студент Самакаев Д.И. группы 8304

Тема практики: Топологическая сортировка

Задание на практику:

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

Алгоритм: Топологическая сортировка (алгоритм Тарьяна):

Сроки прохождения практики: 29.06.2020 – 12.07.2020

Дата сдачи отчета: 12.07.2020

Дата защиты отчета: 12.07.2020

Студент	_____	Бутко А.М.
Студент	_____	Нам Ё Себ
Студент	_____	Самакаев Д.И.
Руководитель	_____	Размочаева Н.В.

АННОТАЦИЯ

В основе учебной практики стоит реализация визуализации алгоритма топологической сортировки на языке программирования Java. Итеративная командная работа обеспечивает обмен опытом как студентов между собой, так и с преподавателем, имеющим реальные практические знания по командной разработке коммерческого продукта. Важной особенностью учебной практики и реализации алгоритма является распределение времени, знаний и умений между всей командой разработки, поиск сильных черт студентов для дальнейшего профессионального и личностного роста.

SUMMARY

The practice is based on the implementation of visualization of the topological sorting algorithm in the Java programming language. Iterative teamwork provides an exchange of experience both among students and with a teacher who has real practical knowledge of team development of a commercial product. An important feature of educational practice and the implementation of the algorithm is the distribution of time, knowledge and skills between the entire development team, the search for the strengths of students for further professional and personal growth.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Целью данной работы является разработка графического интерфейса и визуализации алгоритма топологической сортировки на языке программирования Java, при помощи кооперации группы студентов для решения поставленной задачи. Так же преследовалась цель обновить и закрепить знания по учебным дисциплинам: «Алгоритмы и структуры данных», «Построение и анализ алгоритмов», «Объектно ориентированное программирование», «Дискретная математика» и др.

1. ТРЕБОВАНИЯ К ПРОГРАММЕ

1.1. Исходные Требования к программе

1.1.1. Требования к вводу исходных данных

Ввод исходных данных должен осуществляться следующим образом:

1.1.1.1. Пользователь вводить строку вида: ({*Список вершин через запятую*}, {*Ребра ориентированная графа через запятую*}), где ребра ориентированная графа имеют вид (исток, сток).

1.1.1.2. Пользователь вводит путь до файла, в котором содержится строка аналогичной п.1.1.1.1.

1.1.1.3. Пользователь выбирает режим «песочницы», в котором он может добавить на лист вершины, соединить их между собой, удалить вершины. При окончании создания пользователь нажимает «пуск» и алгоритм начинает работу.

1.1.2. Требование к выводу данных

Пользователь должен видеть поэтапную работу алгоритма, и после ее завершения, отсортированный граф.

1.1.3. Требование к алгоритму

Для топологической сортировки, должен использоваться алгоритм Тарьяна.

1.1.4. Требования к визуализации

Отображение ориентированного графа, отображение переходов по ребрам и покраски вершин, однозначный вывод для считывания пользователем отсортированного графа.

1.2. Уточнение требований после сдачи прототипа

1.2.1. Реализовать работу всех кнопок интерфейса

1.2.2. Сделать вывод программы и корректный выход из нее.

1.2.3. Исправить баги связанные с GUI

1.3. Уточнение требований после сдачи 1-ой версии

1.3.1. Сделать ввод имени вершин при их добавлении

1.3.2. Нарисовать стрелки на ребрах, для видимости ориентирования

1.3.3. Сделать пошаговый проход по алгоритму и возможности возвращения на предыдущий шаг

1.4. Уточнение требований после сдачи 2-ой версии

1.4.1. Добавление диалоговых окон с сообщениями

1.4.2. Добавление функций во вкладке «Помощь»

1.4.3. Обработка ошибки «Цикл»

1.4.4. Добавление плеера

1.4.5. Исправление багов и недочетов в отчете

1.5. План разработки

1.5.1. Обсуждение задания, распределение ролей, выбор средств разработки и структуры данных (Выполнить до 2.07.2020)

1.5.2. Разработка прототипа и первой версии программы (Выполнить до 6.07.2020)

1.5.2.1. Разработка способа хранения данных

1.5.2.2. Разработка способа ввода исходных данных

1.5.2.3. Разработка алгоритма Тарьяна

1.5.2.4. Разработка вывода результирующих данных

1.5.2.5. Тестирование основных компонентов программы (п.2.1.1.1-2.1.1.4.)

1.5.2.6. Разработка визуализации программы

1.5.2.7. Тестирование п.2.1.1.6.

1.5.2.8. Сборка и тестирование программы

1.5.3. Корректировка работы и визуализации программы, т. е. версии 2.0, 3.0 и так далее (Выполнить до 8.07.2020)

1.5.4. Подведение итогов и исправление мелких недочетов (корректировка дизайна, оптимизация алгоритма и т. п.) (Выполнить до 10.07.2020)

1.5.5. Создание финальной версии программы и написание отчет (Выполнить до 10.07.2020)

1.6. Распределение ролей в бригаде

Самакаев Д.М. – Тестировщик и алгоритмист

Бутко А.М. – Фронтенд и алгоритмист

Нам Ё Себ – Документация и алгоритмист

3. ОСОБЕННОСТИ РЕАЛИЗАЦИИ

3.1. Использованные структуры данных

Описание реализованных структур данных указанны в таблице№1.

Таблица №1 – Описание классов и структур данных

Имя класса	Описание
Vertex	Данный класс представляет собой класс вершин графа. Содержит в себе поля: 1. private Point point – координата вершины. 2. private final String name – имя вершины. 3. private Color color – цвет вершины. 4. private final ArrayList<Vertex> vNext – список смежных вершин. 5. public double x, y – Поля для хранения координат Методы: 1. public Vertex(String name) – конструктор принимающий имя вершины. 2. public String getName() – метод возвращающий имя вершины. 3. public void addVNext(Vertex vertex) – метод добавляющий смежную вершину. 4. public void removeVNext(Vertex vertex) – метод удаление смежной вершины 5. public ArrayList<Vertex> getVNext() – метод возвращающий список смежных вершин вершины 6. public double getX(), public double getY() – методы возвращающие координаты x и y вершины 7. public void setPoint(Point point) – метод устанавливающий новые координаты вершины. 8. public void changeColor() – меняет цвет вершины. 9. public Color getColor() – возвращает цвет вершины.
Edge	Данный класс содержит себе ребра графа. Содержит в себе поля: 1. private Vertex vSource, private Vertex vStock – поля вершин (истока и стока) Методы: 1. public Edge(Vertex vSource, Vertex vStock) – конструктор, принимающий вершины (исток и сток). 2. public void vSetSource(Vertex Source), public void vSetStock(Vertex Stock) – методы устанавливающие вершины

	ребра (исток и сток) 3. public Vertex vGetSource(), public Vertex vGetStock() – методы возвращающие вершины (исток и сток) ребра.
States	Данный класс содержит в себе состояния графа. Содержит в себе поля: 1. private ArrayList<ArrayList<Color>> colors – список состояний 2. private int curIndex – текущий индекс состояния Методы: 1. public States() – конструктор класса 2. public void addState(ArrayList<Color> colorState) – метод добавления текущего состояния графа 3. public ArrayList<Color> getState() – метод получения текущего состояния. 4. public ArrayList<Color> nextState() – метод переключения на следующее состояние 5. public ArrayList<Color> prevState() – метод переключения на предыдущее состояние 6. public ArrayList<Color> setStartState() – метод устанавливающий начальное состояние графа 7. public ArrayList<Color> setFinishState() – метод устанавливающий конечное состояние графа
Digraph	Данный класс содержит в себе граф. Содержит в себе поля: 1. private Map<String, Vertex> graph – словарь, по которому строится сам граф. 2. private ArrayList<Edge> edges – список ребер графа 3. public States states – список состояний графа Методы: 1. public Digraph() – конструктор класса. 2. public boolean isEmpty() – метод проверяющий граф на пустоту. 3. public Map<String, Vertex> getMap() – получение словаря графа 4. public Vertex getElement(String key) - метод получения элемента графа по ключу 5. public void addVertex(String name) – метод добавления вершины в граф 6. public void addEdge(String vFrom, String vTo) - метод добавления ребра в граф через задание двух точек. 7. public void addEdge(Edge edge) – метод добавления ребра в граф через задание ребра. 8. public void removeEdge(Vertex vertexFrom, Vertex

	vertexTo) – метод удаления ребра из графа. 9. public void removeVertex(Vertex vertex) – метод удаления вершины графа. 10. private boolean removeIncEdge(Vertex vertex) – метод удаление смежных ребер графа указанной вершины. 11. public ArrayList<Edge> getEdges() – метод получения списка ребер public void gHasError(ErrorType e) – метод указывающий на ошибку в графе.
Algorithm	Класс алгоритма Тарьяна. Содержит в себе переменные: 1. private final Stack<Vertex> vStack – стек, используемый в алгоритме сортировки 2. private final Digraph digraph – сортируемый граф 3. private ArrayList<String> sortResult – список, хранящий в себе результат сортировки

3.2. Основные методы

Основные методы для работы были реализованы в классе Algorithm, который реализует топологическую сортировку алгоритмом Тарьяна.

Реализованные основные методы описаны в таблице №2.

Таблица № 2 – Описание основных методов программы

Объявление метода	Описание
public Algorithm (Digraph digraph);	Конструктор, принимающий граф, к которому будет применяться алгоритм
private void addColorsState();	Метод который добавляет новое состояние графа в список состояний, для того, чтобы можно было показать ход работы алгоритма
private void helpSort(Vertex vCurrent);	Вспомогательный рекурсивный метод (поиск в глубину графа) для топологической сортировки, он принимает текущую вершину, с которой работаем, проверяем ее

	цвет, если серая следовательно наш граф имеет цикл, значит топологическая сортировка невозможна (будет некорректной), если цвет белый, следовательно меняет цвет вершины на серый, и идем по следующим смежным вершинам (потомкам), при выходе из итерации рекурсии, вершина закрашивается в черный и записывается в стек, при этом каждое изменение цвета вершины, записывается в список состояний графа.
public ArrayList<String> sort();	Головной метод сортировки, он запускает вспомогательный рекурсивный метод helpSort(), который делает поиск в глубину, после выхода из рекурсии, метод записывает отсортированный граф в строку, посредством снимания элемента со стека, и возвращает ее.

4. ТЕСТИРОВАНИЕ

4.1. Ручное тестирование

4.1.1. Тестирование запуска программы.

Запуск программы (см. рис. 1)

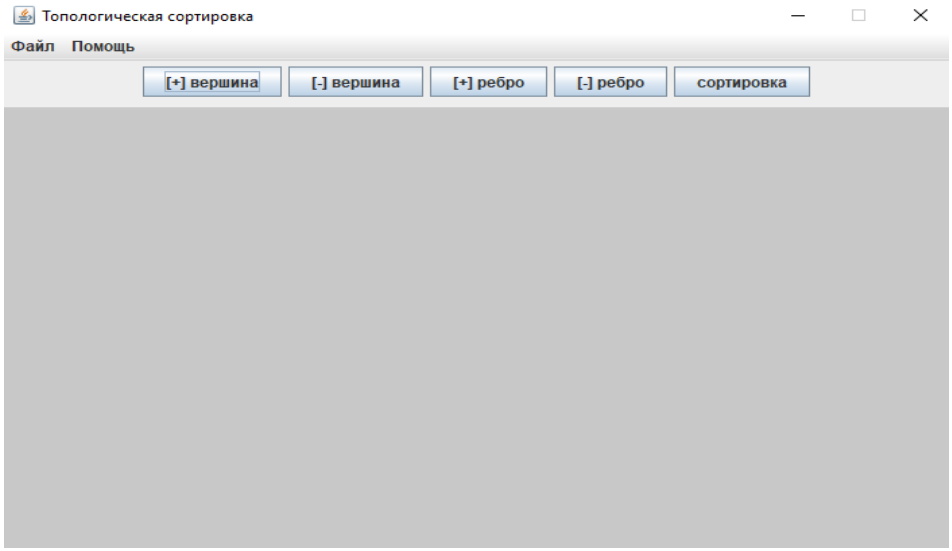

Рисунок 1 – Запуск программы

4.1.2. Тестирование ввода.

При нажатии кнопки «[+] вершина» создается вершина. В случае присваивания новой вершине, существующего имени, появляется окно с сообщением (см. рисунок 2).

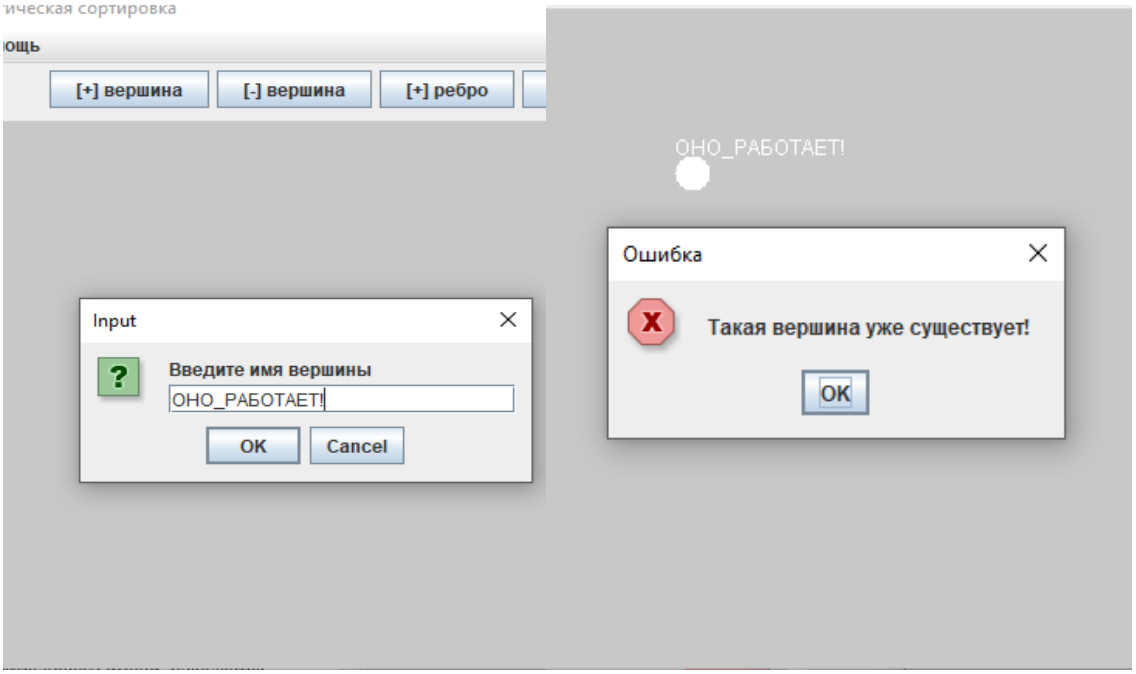

Рисунок 2 – Создание вершин графа

4.1.3. Тестирование создания/удаления ребер

При нажатии «[+] вершина»/ «[-] вершина» нужно указать две вершины, чтобы создать ребро графа (см. рис. 3), удаление происходит аналогичным способом (см. рис. 3).

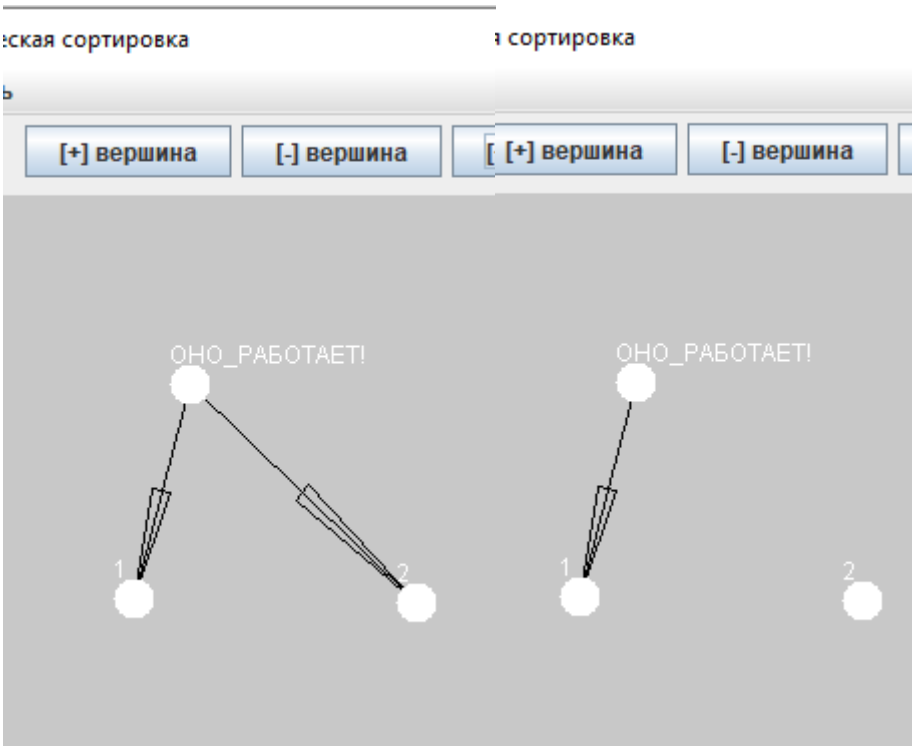

Рисунок 3 – Создание и удаление ребра

4.1.4. Тестирование удаления вершины

При нажатии на «[+] вершина» нужно указать на существующую вершину, должны удалиться смежные вершины (см. рис. 4)

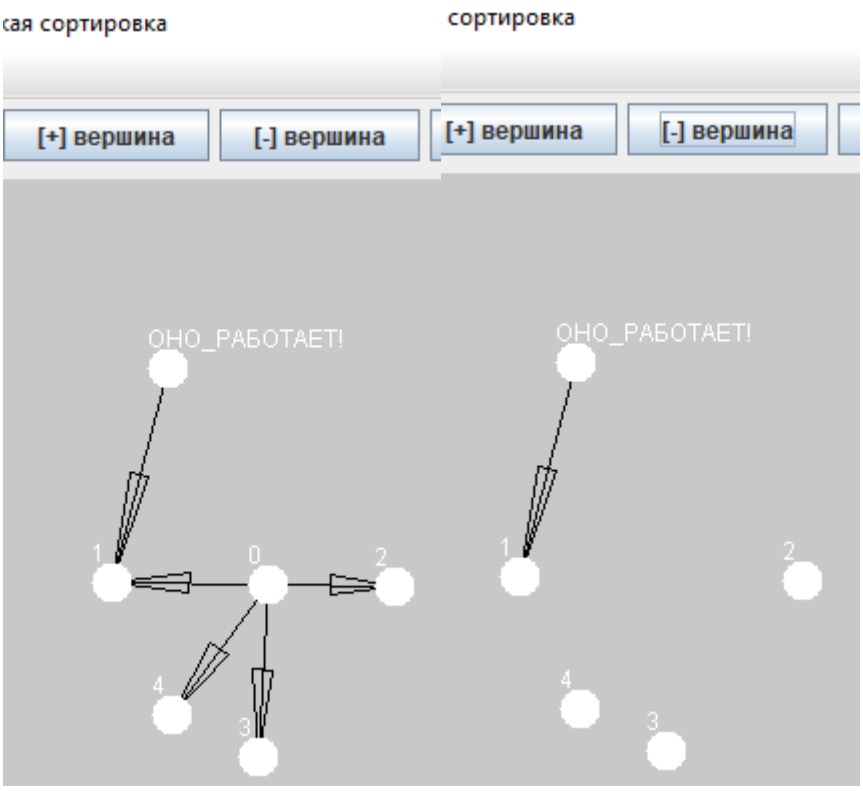

Рисунок 4 - Удаление вершины

4.1.5. Тестирование сохранения/открытия в/из файла.

При нажатии на сохранить/открыть должен сохраниться/открыться нарисованный пользователем граф (см. рис. 5).

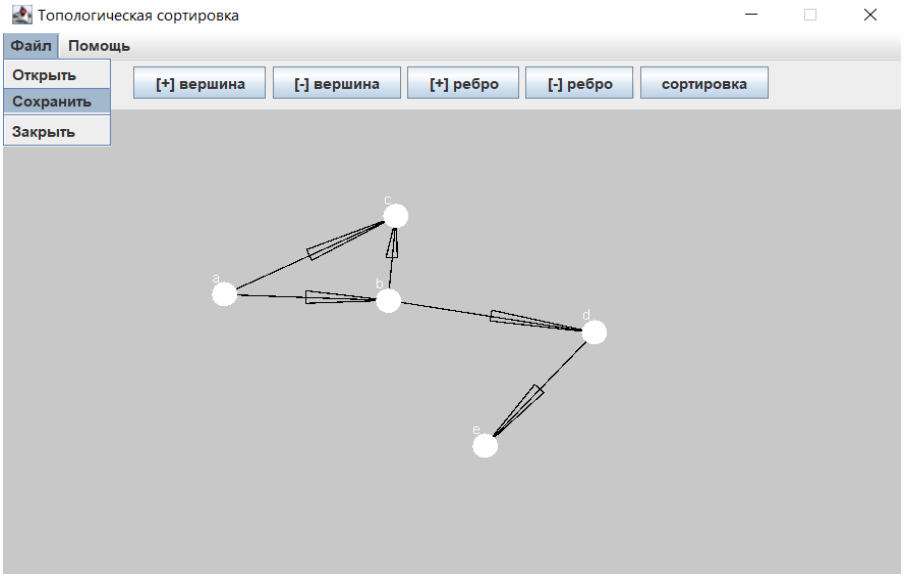

Рисунок 5 – Сохранение/открытие в/из файла

4.1.6. Тестирование результатов работы алгоритма.

При нажатии на кнопку «сортировка», в информационном окне выводится результат в виде строки (см. рис. 6).

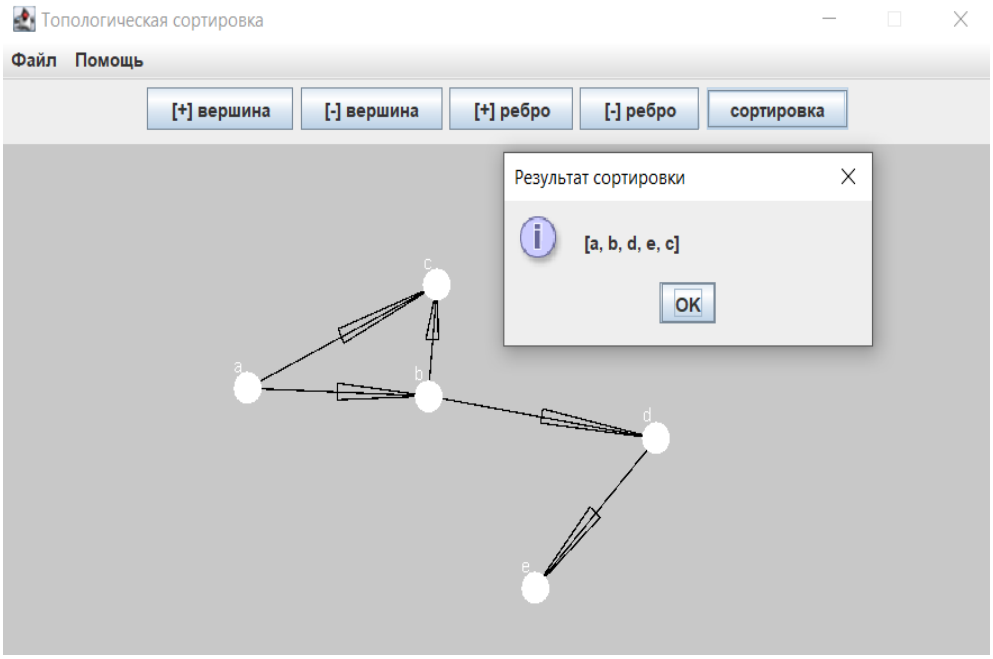

Рисунок 6 – Результат работы алгоритма в виде строки.

4.1.7. При нажатии на кнопку «ОК», программа выведет результат работы алгоритма в виде графа (см. рис. 7).

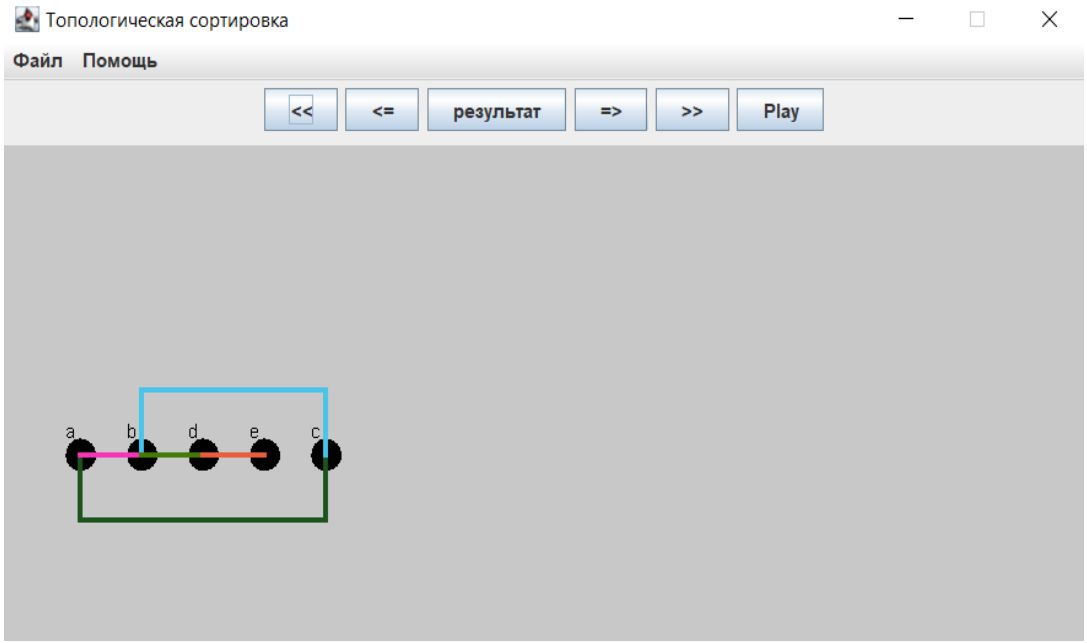

Рисунок 7 – Результат работы алгоритма в виде графа.

4.1.8. При нажатии на «Запуск», запустится анимированная демонстрация работы алгоритма (см. рис. 8).

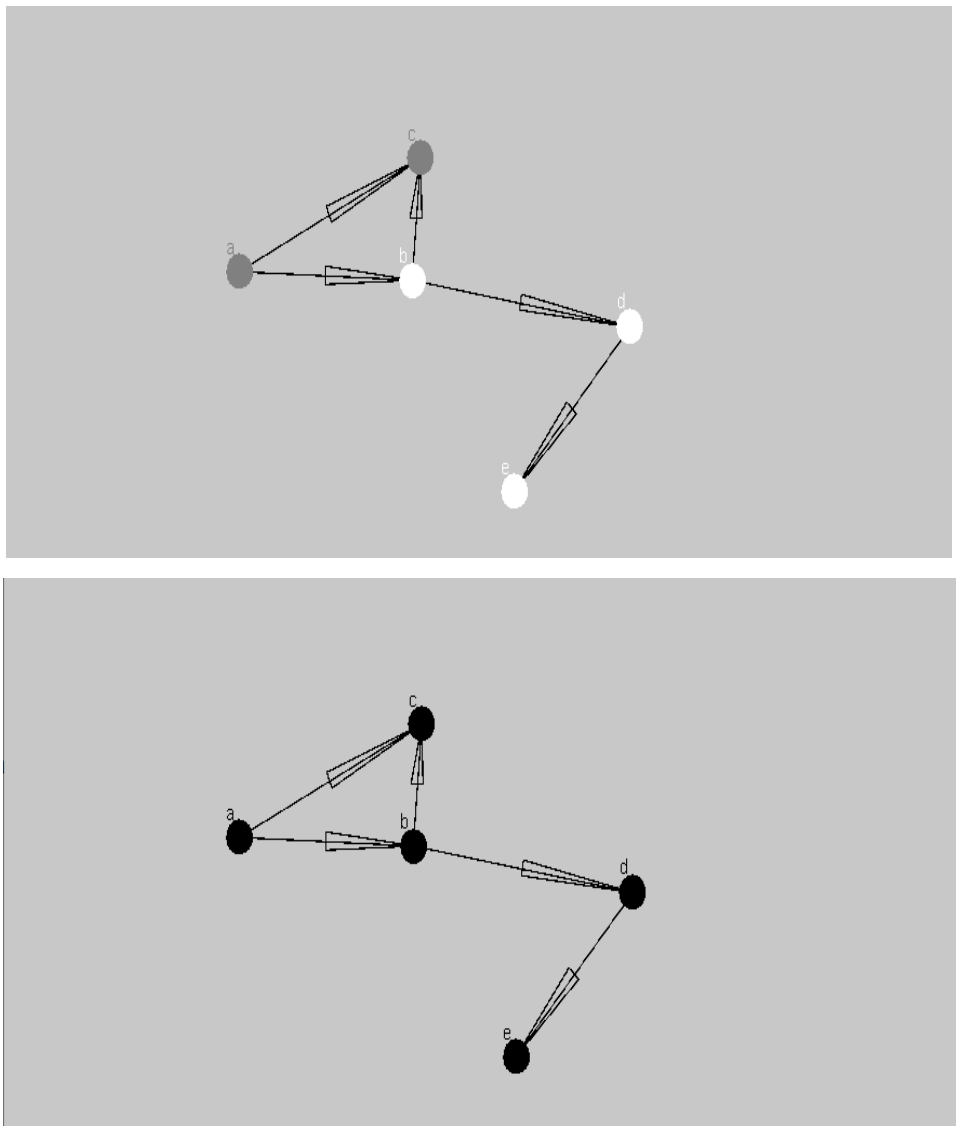

Рисунок 8 – Анимация работы алгоритма

4.1.9. Для перемещения между состояниями графа были реализованы соответствующие методы, привязанные к кнопкам: «<<» - в начало, «>>» - в конец, «<=» - шаг назад, «=>» - шаг вперёд (см. рис. 9).

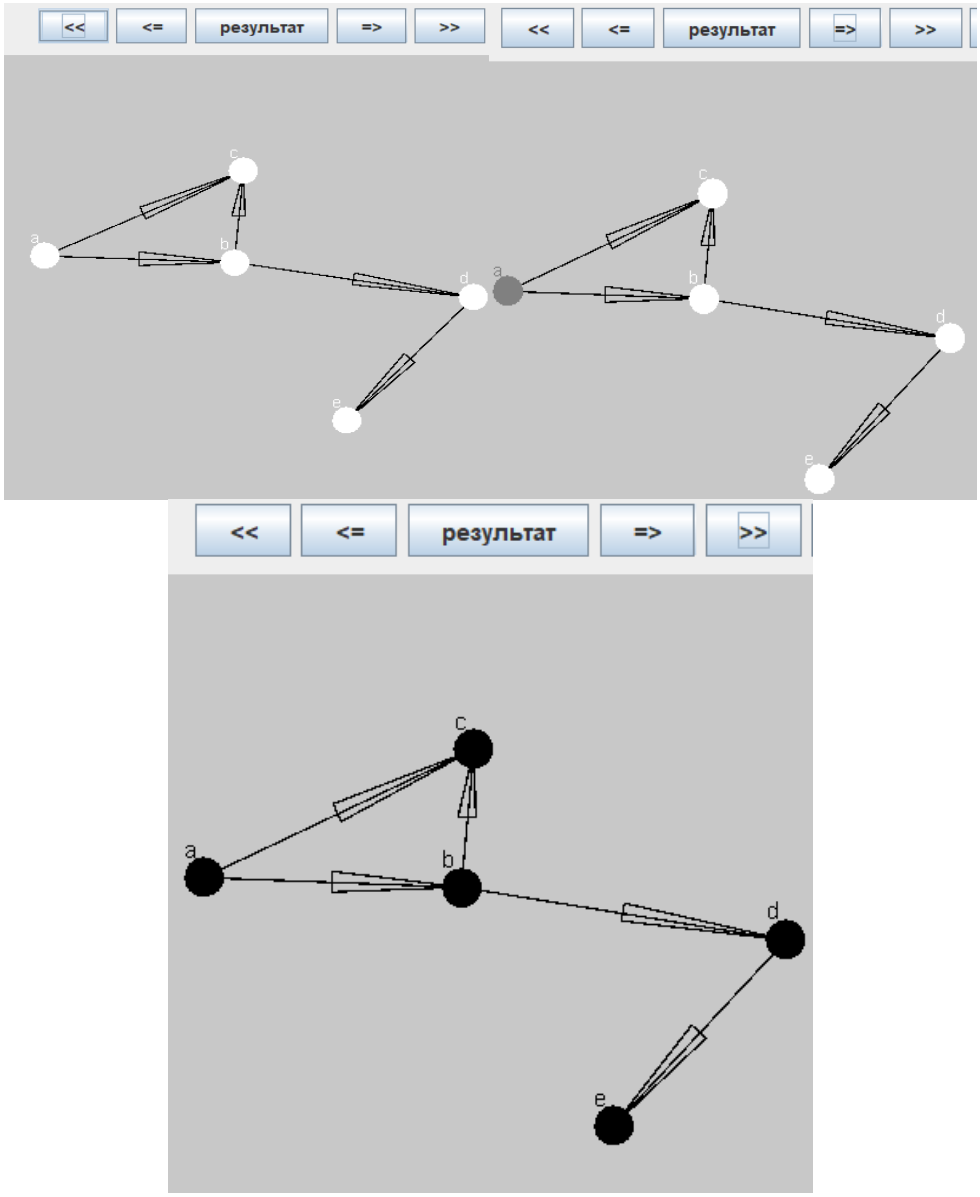

Рисунок 9 – Кнопки в начало, в конец, вперед, назад.

ЗАКЛЮЧЕНИЕ

Разработка поставленной задачи была выполнена в соответствии с планом. Было спроектировано и реализовано приложение для топологической сортировки графа (алгоритм Тарьяна). Был реализован графический интерфейс, визуально отображающий результаты работы алгоритма и позволяющий управлять возможностями приложения. Приложение было протестировано вручную.

Таким образом разработка приложения была завершена успешно с полным выполнением плана.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. ГОСТ 7.32-2017 Система стандартов по информатизации, библиотечному и издательскому делу. Отчет о научно-исследовательской работе. Структура и правила оформления.

2. Учебный курс https://stepik.org/course/187/syllabus

3. Java Platform, Standard Edition & Java Development Kit Version 9 API Specification https://docs.oracle.com/javase/9/docs/api/overview-summary.html

4. Java Platform, Standard Edition 8 API Specification https://docs.oracle.com/javase/8/docs/api/

5. Репозиторий разработанного проекта https://github.com/artbutko/SummerPractice.TopologicalSort