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. 0303 Торопыгин А.С.

Студент гр. 0304 Шквиря Е.В.

Руководитель Фирсов М.А.

Санкт-Петербург
2022

ЗАДАНИЕ

НА УЧЕБНУЮ ПРАКТИКУ

Студент Торопыгин А.С. группы 0303

Студент Шквиря Е.В. группы 0304

Тема практики: Топологическая сортировка

Задание на практику:

Командная итеративная разработка визуализатора алгоритма на Kotlin с графическим интерфейсом.

Алгоритм: Топологическая сортировка.

Сроки прохождения практики: 29.06.2020 – 12.07.2020

Дата сдачи отчета: 12.07.2020

Дата защиты отчета: 12.07.2020

Студент		Торопыгин А.С.
Студент		Шквиря Е.В.
Руководитель		Фирсов М.А.

АННОТАЦИЯ

Требуется разработать программу, визуализирующую выполнение алгоритма топологической сортировки на произвольном графе. В качестве дополнительного функционала выступает пошаговое выполнение и считывание и сохранение данных в файл. Алгоритм топологической сортировки используется для упорядочивания вершин в графе.

Целью работы является формирование навыков командной работы, изучение новых языков программирования и работы с фреймворками.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Целью практической работы является разработка программы, визуализирующей работу алгоритма топологической сортировки на графе.

1. ТРЕБОВАНИЯ К ПРОГРАММЕ

1.1. Исходные Требования к программе

1.1.1. Требования к визуализации

Приложение должно иметь понятный графический интерфейс. Он должен полностью описывать функционал программы. Должна быть возможность при помощи мыши выбирать место создания и удаления элементов для редактирования графа. Должно быть пространство с выводом текстовой информации о выполнении алгоритма (этапы алгоритма, актуальное состояние, промежуточные шаги).

1.1.2. Требования к вводу исходных данных

Исходные данные поступают в приложение посредством считывания с файла или созданием графа вручную пользователем (при помощи графического интерфейса).

1.1.3. Требования к структуре программы

Явное разделение программы на несколько слоёв: слой данных, слой бизнес-логики, слой отображения.

1.1.4. Требования к языку

Написание программы на языке программирования Kotlin с использованием Kotlin Multiplatform и инструментов Jetpack Compose.

1.1.5. Требования к тестированию

Тестирование программы разделено на ручное и автоматическое. Вручную будут тестироваться элементы графического интерфейса (создание графа пользователем, передвижение по алгоритму), автоматически будет тестироваться часть, скрытая от пользователя (считывание из файла и сохранение в файл данных, создание объектов классов, алгоритм). Автоматическое тестирование будет реализовано при помощи UNIT-тестов.

1.2. Уточнение требований

1.2.1. Требования к визуализации

Для прототипа первой версии программы был составлен эскиз интерфейса (см. рис. 1).

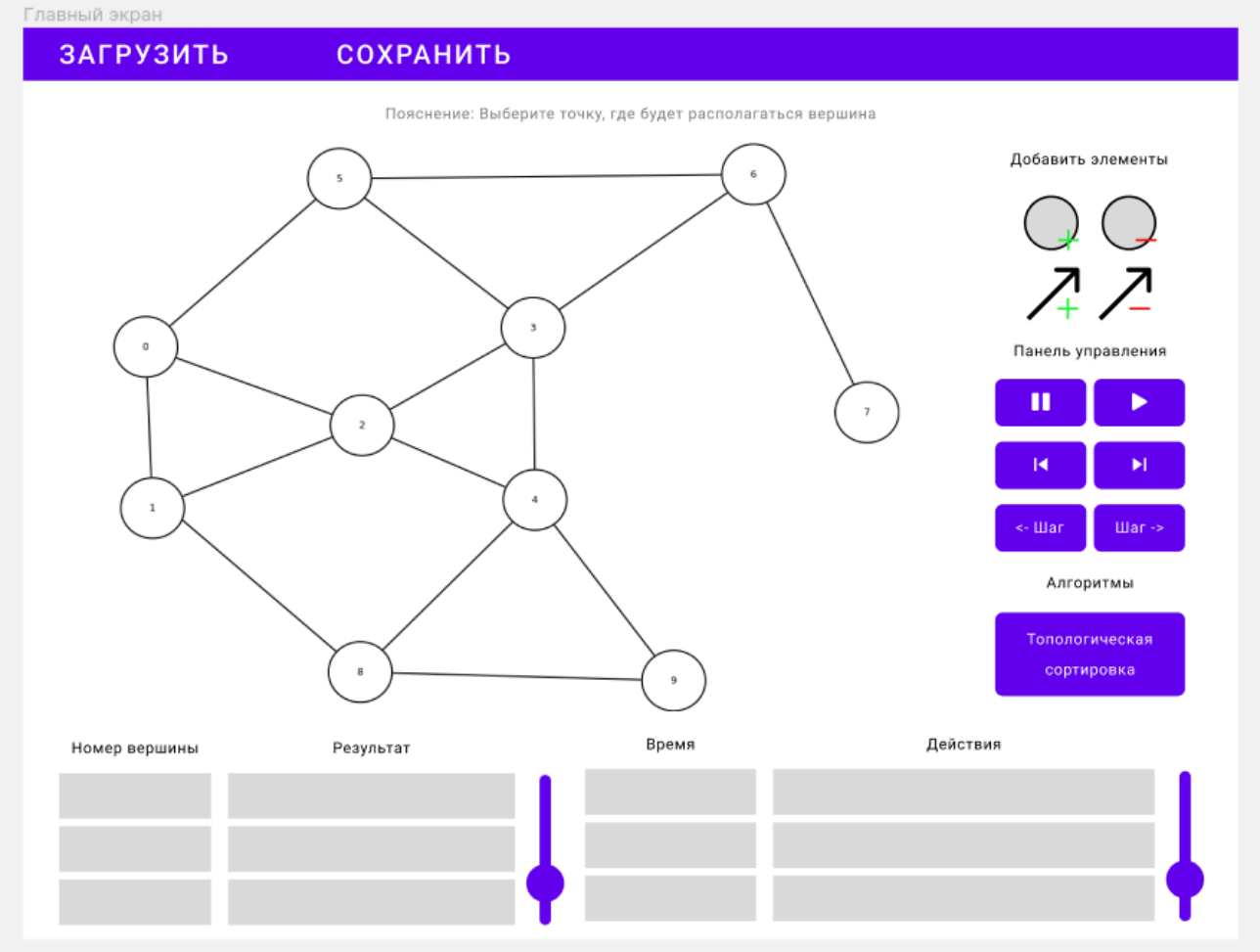

Рис. 1 - Эскиз прототипа программы

На верхней панели инструментов расположены две кнопки - загрузить и сохранить. Первая позволяет открывать для пользователя окно выбора файла, и в случае, если формат файла поддерживается, начинает процедуру считывания информации. В качестве допустимого формата представления данных был выбран JSON. Вторая сохраняет информацию о графе в файл. Способ представления будет описан ниже.

В поле “Пояснение” будет выводиться информация о необходимом действии, которое ожидается от пользователя.

Главная часть экрана приложения будет содержаться холст для работы с графом. На нём будет графическое представление данных в формате отображения вершин и рёбер. Справа от холста располагается панель инструментов для работы с графом - добавление/удаление вершины, добавление/удаление ориентированного ребра. При добавлении вершины на холст сканируется пространство для определения, не закроет ли эта вершина другую вершину, и поместится ли она полностью на холст. После этого откроется окно для ввода названия вершины. При добавлении ребра происходит выбор двух вершин, после чего происходит соединение. Первая при выборе окрашивается в жёлтый цвет. После добавления ребра цвет обоих вершин чёрный. При удалении вершины все связанные с ней рёбра тоже удаляются. При удалении ребра происходит разрыв связи между двумя вершинами. Также при удалении ребра первая вершина окрасится в красный. При удалении ребра его ориентация значения не имеет. После удаления обе вершины также будут иметь чёрный цвет.

Также на панели инструментов находится кнопка для старта работы алгоритма топологической сортировки. Кнопка “Топологическая сортировка” запускает алгоритм топологической сортировки, который работает в автоматическом режиме с некоторой задержкой между шагами алгоритма (по умолчанию, 0.5 секунд между двумя действиями). При нажатии кнопки “Шаг вперёд” или “Шаг назад” управление программой переходит пользователю и он может перемещаться между состояниями алгоритма самостоятельно. При нажатии кнопки “Пауза” программа останавливается на текущем шаге алгоритма и ждёт дальнейших указаний к работе. При нажатии кнопки “Продолжить” алгоритм продолжает свою работу в автоматическом режиме. Также есть две кнопки: “к началу” и “к концу” алгоритма. Первая осуществляет переход к началу работы алгоритма, вторая - к концу. Если в данный момент выполнялась автоматическая визуализация алгоритма - перемещение в начало прерывает её.

Также при нажатии на элемент добавления/удаления вершины/ребра кнопки для запуска алгоритмов и работы с ними блокируются, чтобы не нарушать их работу. Аналогично и с кнопками алгоритма, при старте алгоритма редактирование графа блокируется.

Ниже холста располагаются таблицы с текстовой информацией о работе алгоритма. Левая таблица отвечает за результат работы, правая - за промежуточные шаги. В первой таблице: в левой части располагается номер вершины, в правой части - результат работы алгоритма для этой вершины. Во второй таблице: в левой части располагается время, в которое произошло действие. В правой части - описание действия.

1.2.2. Требования к вводу входных данных.

Входные данные хранятся в файле JSON, в котором есть поля:

- name - имя вершины (строка);
- id - номер вершины (десятичное число);
- point - координата центра вершины (два вещественных числа);
- order - номер порядка (десятичное число);
- edges - список названий вершин, в которые идут рёбра из текущей вершины (список строк).

Было принято решение хранить имя и номер вершины. Это даёт несколько преимуществ:

- Это позволит программе работать с графом, в котором некоторые вершины могут иметь одинаковое название;
- В ходе разработки удобнее будет работать с номером вершины, а не с её именем.

1.2.3. Требования к структуре программы

В программе будет присутствовать разбиение архитектуры на несколько слоёв: слой данных, слой бизнес-логики и слой отображения. Явное разбиение

будет выглядеть таким образом: Data, Models и UI. Взаимодействие между слоями приведено на рисунке 2.

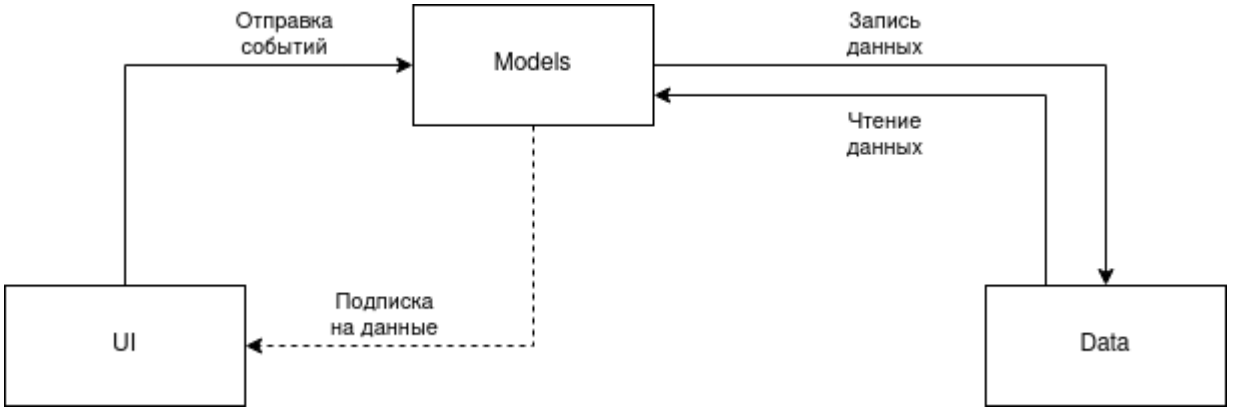

Рис. 2 - Слои архитектуры программы

2. ПЛАН РАЗРАБОТКИ И РАСПРЕДЕЛЕНИЕ РОЛЕЙ В БРИГАДЕ

2.1. План разработки

Таблица 1. Поэтапный план разработки приложения

Дата	Задача	Статус выполнения
04.07.2022	Согласование спецификации	Выполнено
05.07.2022	Добавление классов для хранения данных. Загрузка информации о графе из файла. Функции для расположения на холсте.	Выполнено
06.07.2022	Добавление функционала для UI-элементов (добавление элементов графа). Сдача прототипа	Выполнено
07.07.2022	Добавление функционала для UI-элементов (удаление элементов графа). Реализация алгоритма топологической сортировки.	Выполнено
08.07.2022	Сохранение информации о графе. Добавление проверки валидности графа. Сдача 1-й версии программы	Выполнено
09.07.2022	Реализация разбиения работы топологической сортировки на состояния. Передача информации о шагах в таблицу логирования.	Выполнено
10.07.2022	Передача информации о шагах в UI. Добавление функционала шагов вперёд/назад. Проверка валидности данных при считывании из файла. Сдача 2-й версии программы	Выполнено
11.07.2022	Подготовка релизной версии проекта. Внесение корректировок.	Выполнено

	Тестирование.	
12.07.2022	Сдача финальной версии программы	

2.2. Распределение ролей в бригаде

Шквиря Е.В.:

- Создание структур данных (ViewObject)
- Проектирование и реализация графического интерфейса.
- Составление архитектуры приложения, разделение приложения на логические слои.
- Реализация графической части работы алгоритма топологической сортировки, в том числе касающейся отрисовки и получения информации об этапах работы алгоритма.
- Тестирование своей части работы.

Торопыгин А.С.:

- Создание структур данных (Data)
- Реализация алгоритма топологической сортировки.
- Разбиение алгоритма на логические части для возможности пошагового графического отображения работы алгоритма, в том числе логирование этапов работы алгоритма.
- Реализация сохранения и загрузки данных.
- Тестирование своей части работы.

3. ОСОБЕННОСТИ РЕАЛИЗАЦИИ

3.1. Структуры данных

3.1.1. Граф

Для хранения данных были разработаны структуры данных *Vertex* (класс вершины) и *Graph* (класс графа). В классе вершины есть все необходимые поля для хранения вершины: имя, идентификатор, порядок, координаты центра и список исходящих рёбер. Класс графа хранит список вершин.

3.1.2. Отображение графа

Для явного разделения структур данных для хранения в памяти и для отрисовки был создан класс *VertexVO* наподобие *Vertex*. Данный класс хранит в себе поля: идентификатор вершины, название вершины, координаты центра вершине в формате точки и стандартный цвет для отрисовки. Также в этом классе есть статичные константные поля - радиус вершины для отрисовки и максимально допустимое количество символов. При передаче данных из слоя данных в слой отрисовки данные *Vertex* конвертируются в данные *VertexVO* через специально написанный маппер *Vertex.toVertexVO()*.

3.1.3. Сохранение и загрузка графа

Для сохранения и загрузки графа был написан класс *Parser*.

3.1.4. Алгоритм и его логирование

Для работы с алгоритмом был создан объект *GraphAlgorithm*. Чтобы выводить информацию о работе алгоритма на экран, был создан класс *State*, который позволяет отслеживать текущие состояния алгоритма и передавать их на слой визуализации.

3.1.5. Интерфейс

Для обработки состояний нажатий на кнопки панели редактирования графа и управления визуализаций алгоритмов был написан *enum* класс *GraphToolsState*, в котором собраны все состояния изменения отображаемого на холсте. Также от них зависит отображение данных в таблице действий алгоритма.

3.2. Основные методы

3.2.1. Граф

Для классов *Graph* и *Vertex* были написаны различные геттеры для получения информации о том или ином поле объекта (например, получить имя или идентификатор вершины или получить список вершин из графа.)

3.2.2. Отображение графа

Для отрисовки графа был использован элемент UI - Canvas. Данный виджет позволяет наносить рисунки на свой холст - круг, линию, текст и т.п. Для удобной отрисовки была написана библиотека *CanvasDrawLib*, в которой содержатся публичные методы *drawVertex* и *drawEdge*, получающие на вход холст Canvas и данные для отрисовки вершины и ребра соответственно. Внутри себя они также используют приватные методы, например, для нанесения имени вершины или вычисления координат начала и конца ребра. Для сохранения разделения программы на несколько слоёв был написан класс *GraphCanvasViewModel*, который содержит в себе логику обработки запросов пользователя, а также обновлением данных для их визуализации на холсте. Для поддержания актуального состояния данный класс также содержит подписку на хранилище данных о графе, и в случае его изменения (например,

при загрузке нового графа из файла или добавления необходимости менять цвет вершин), производит отрисовку нового графа.

3.2.3. Сохранение и загрузка графа

Методы класса *Parser* способны считывать и сохранять информацию о графе из файлов типа json при помощи сторонней библиотеки GSON. Для хранения данных графа используется реализованый *singleton*-объект *DataGraphLocator*. В нём есть методы *readGraphData* и *saveGraphData*, для запуска методов *Parser* и обновления состояния графа. *ViewModel`*и элементов UI имеют подписку на данные о графе в этом объекте, поэтому в них всегда загружается актуальное состояние графа.

3.2.4. Алгоритм и его логирование

Для работы алгоритма в объекте *GraphAlgorithm* был описан публичный метод *TopSort.* Для его корректной работы были написаны приватные методы *TopSortUtil, checkGraphForCycle* и т.д. Каждый приватный метод необходим для корректной работы алгоритма.

3.2.5. Интерфейс

Как было описано выше, интерфейс регистрирует запросы на изменение графа или запуск алгоритма через состояния. Для отслеживания состояний использовались *MutableFlow,* доступные из библиотеки *Coroutines*. При создании *ViewModel`*ей в них происходит регистрация на различные состояния, которые необходимо отслеживать для корректной работы элемента UI. Также отдельно стоит отметить, что в приложении используются всплывающие окна, наприме, для отображения имени вершины или ввода имени файла для загрузки/сохранения графа

4. ТЕСТИРОВАНИЕ

4.1. Тестирование графического интерфейса

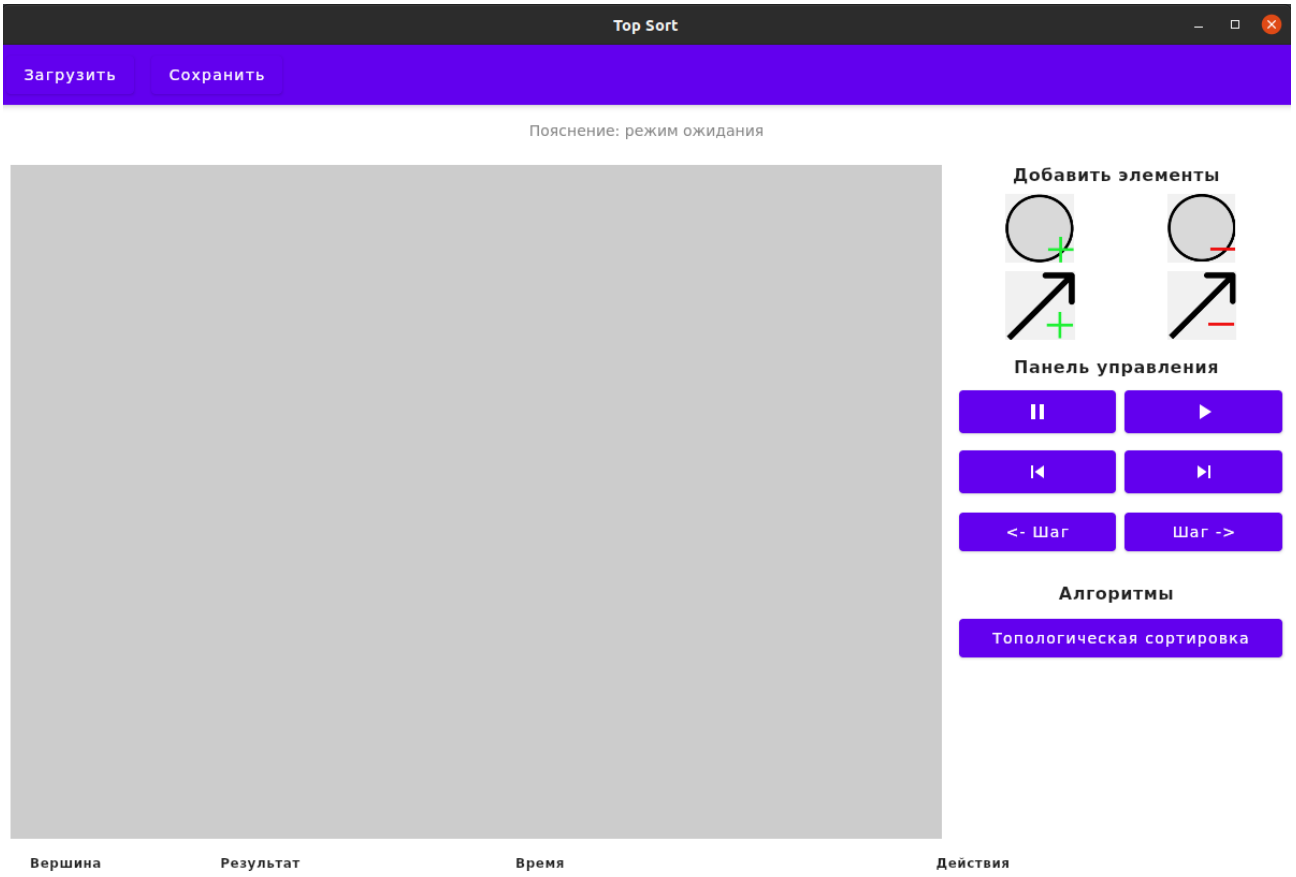

Рис. 3 - Запуск приложения + малый размер окна

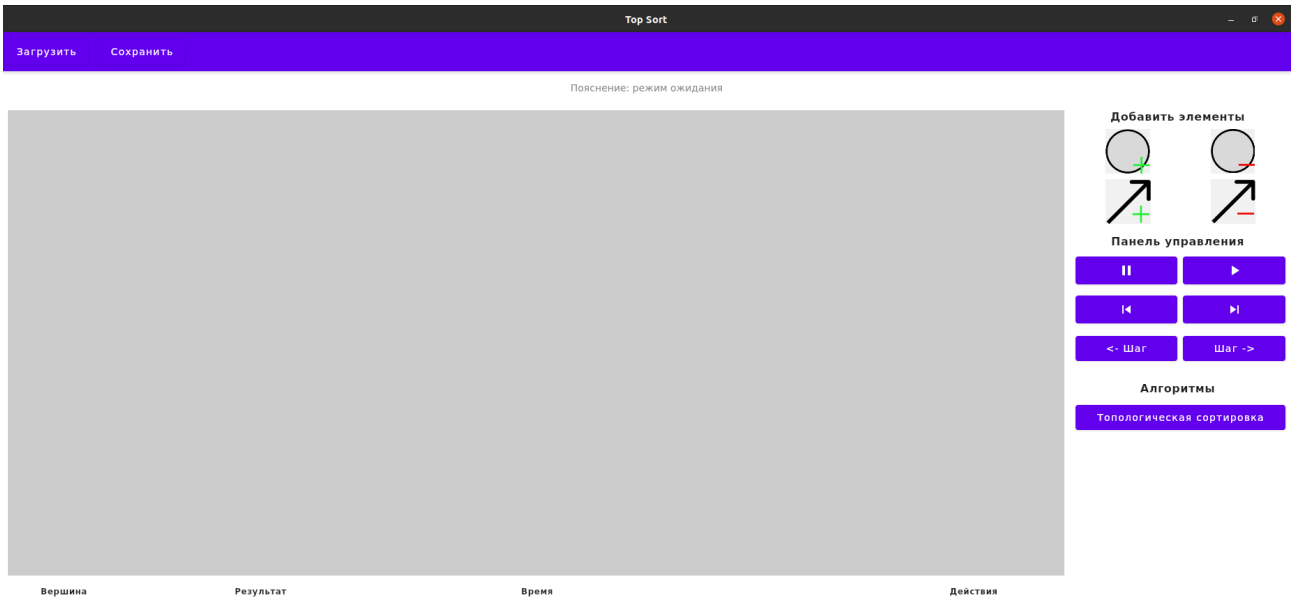

Рис. 4 - Полный размер окна

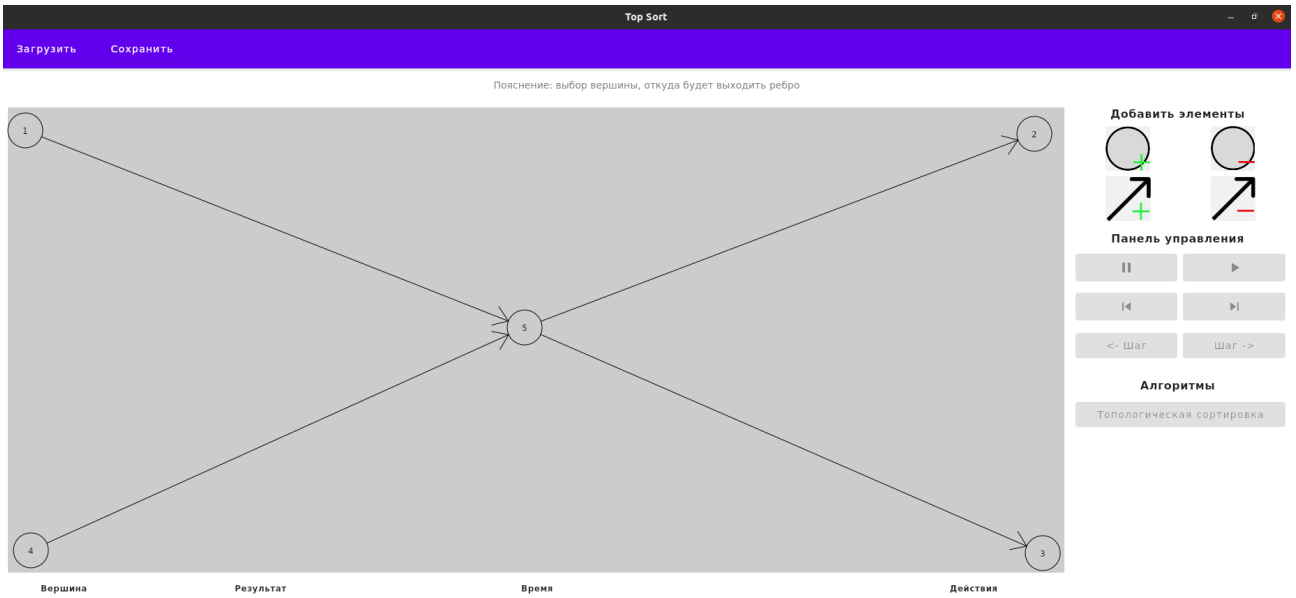

Рис. 5 - Размещение вершин и рёбер в различных местах

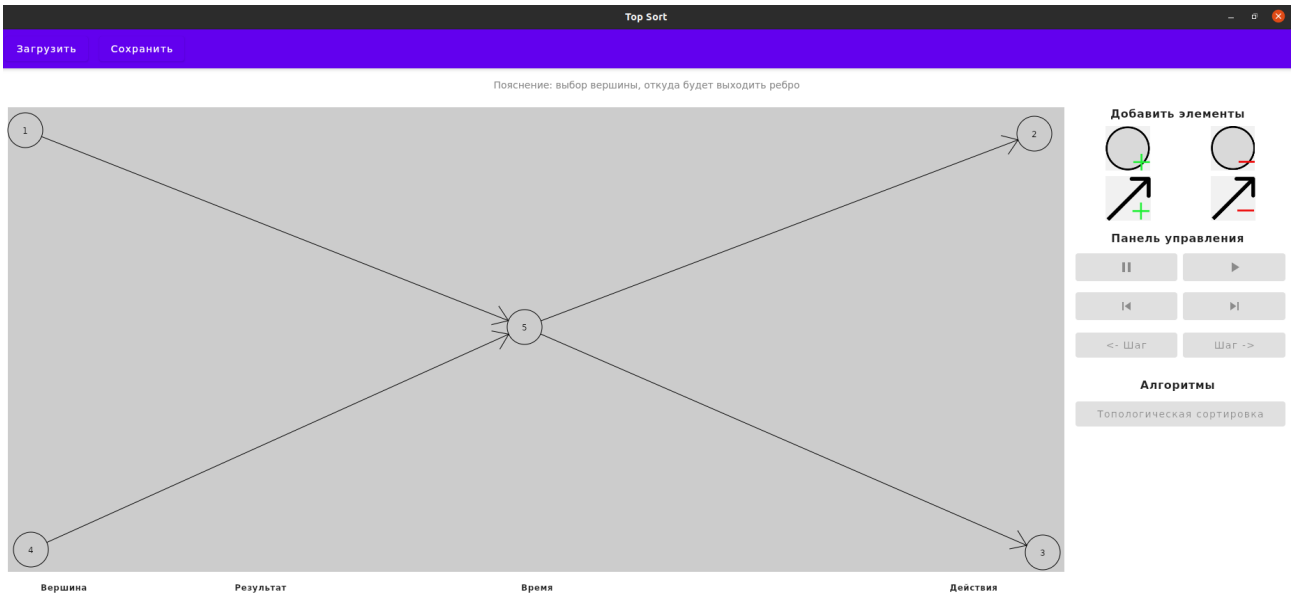

Рис. 6 - Блокировка кнопок алгоритма при редактировании графа

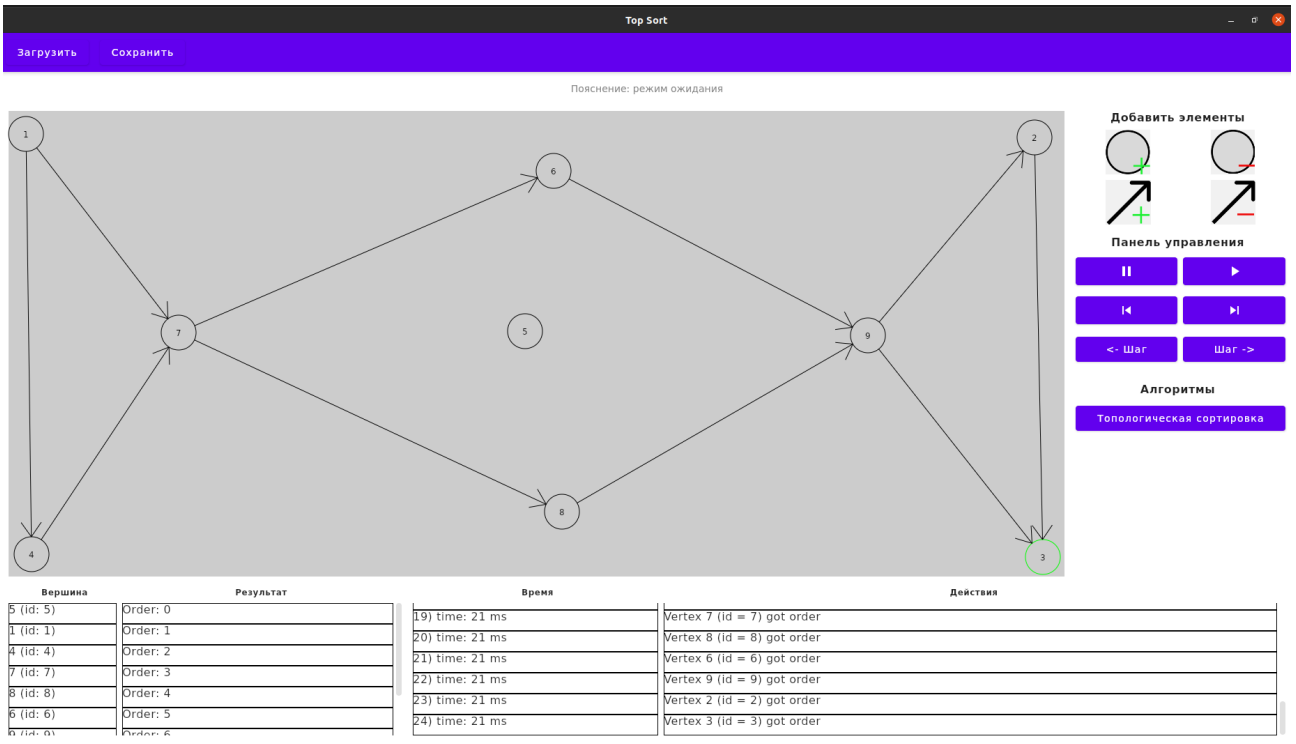

Рис. 7 - Запуск алгоритма

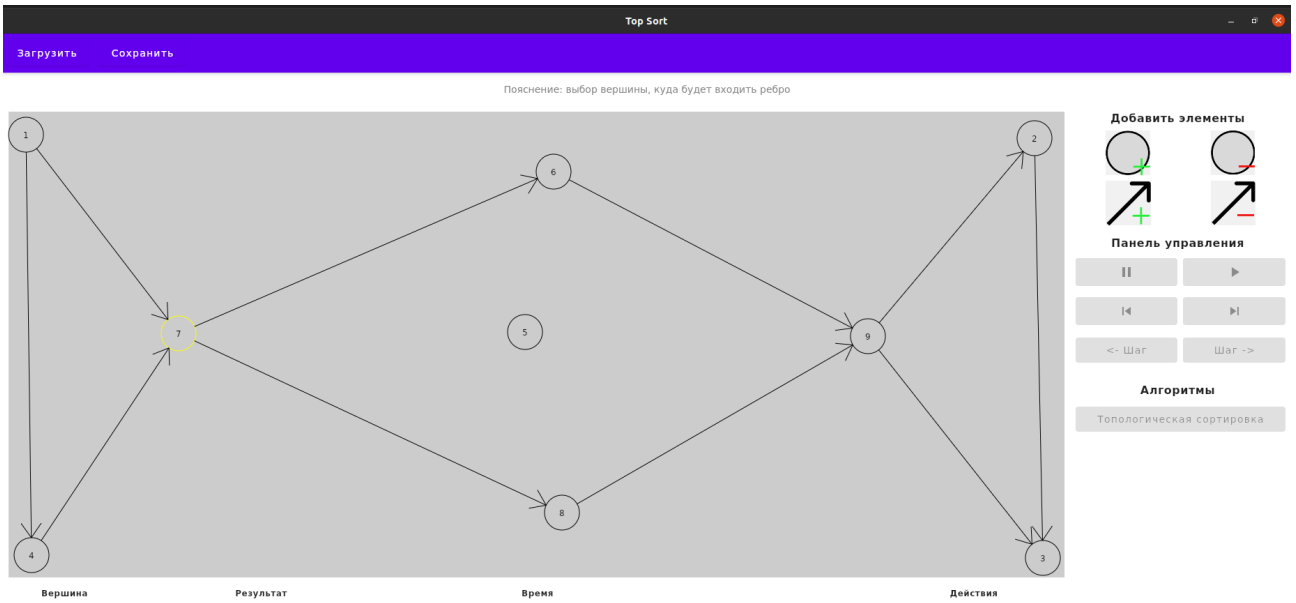

Рис. 8 - Добавление ребра

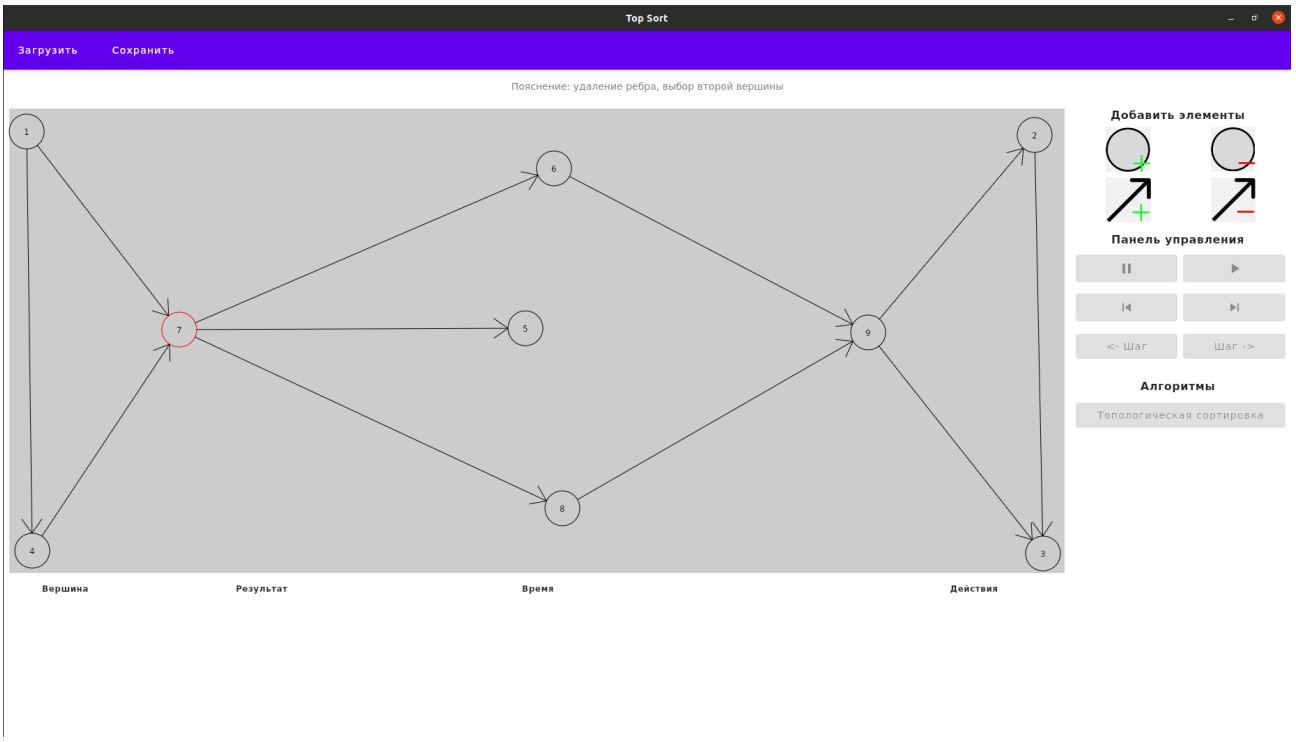

Рис. 9 - Удаление ребра

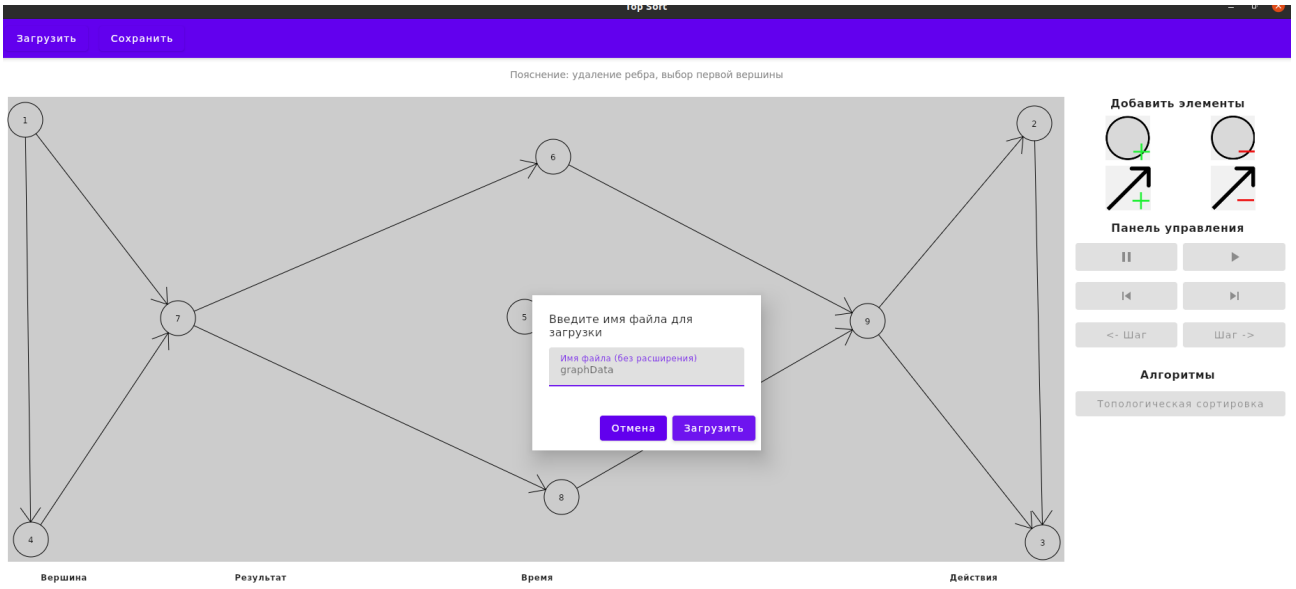

Рис. 10 - Загрузка графа из файла

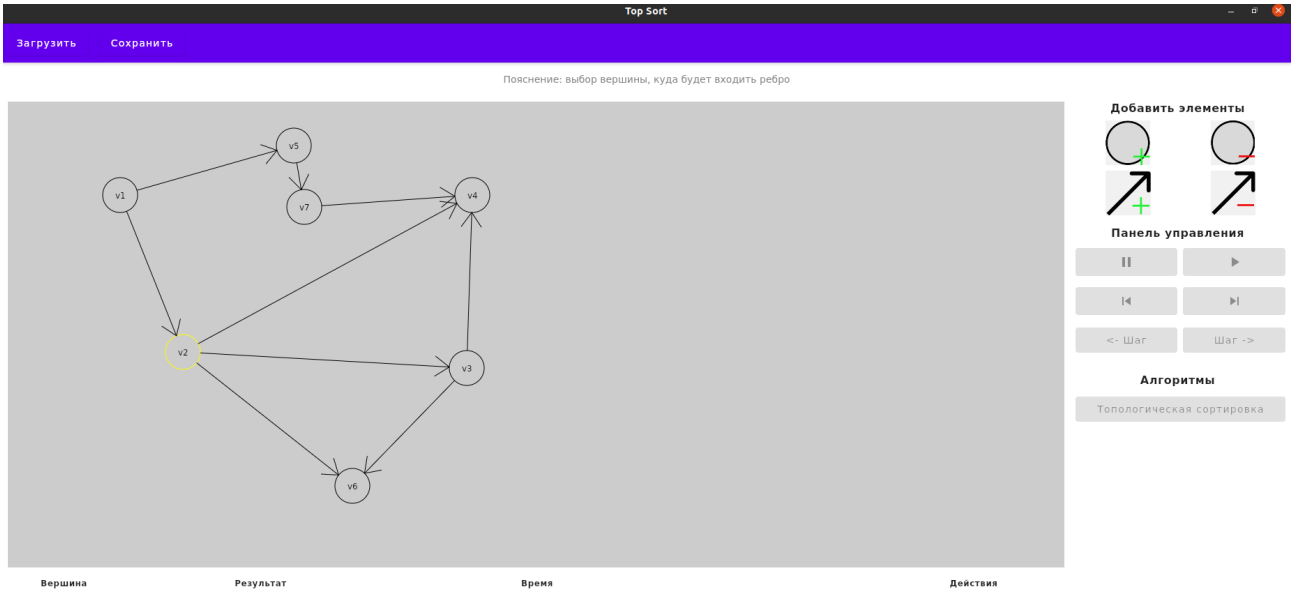

Рис. 11 - Отображение загруженного графа из файла + взаимодействие с ним (добавление ребра от v2)

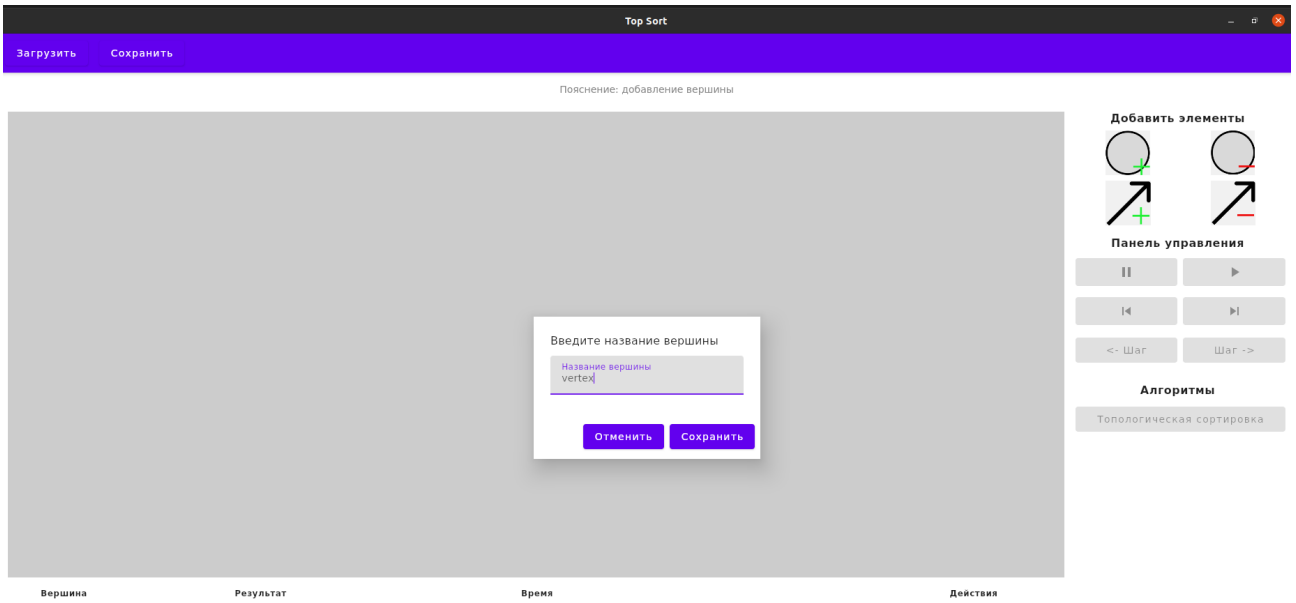

Рис. 12 - Добавление вершины + ввод её имени

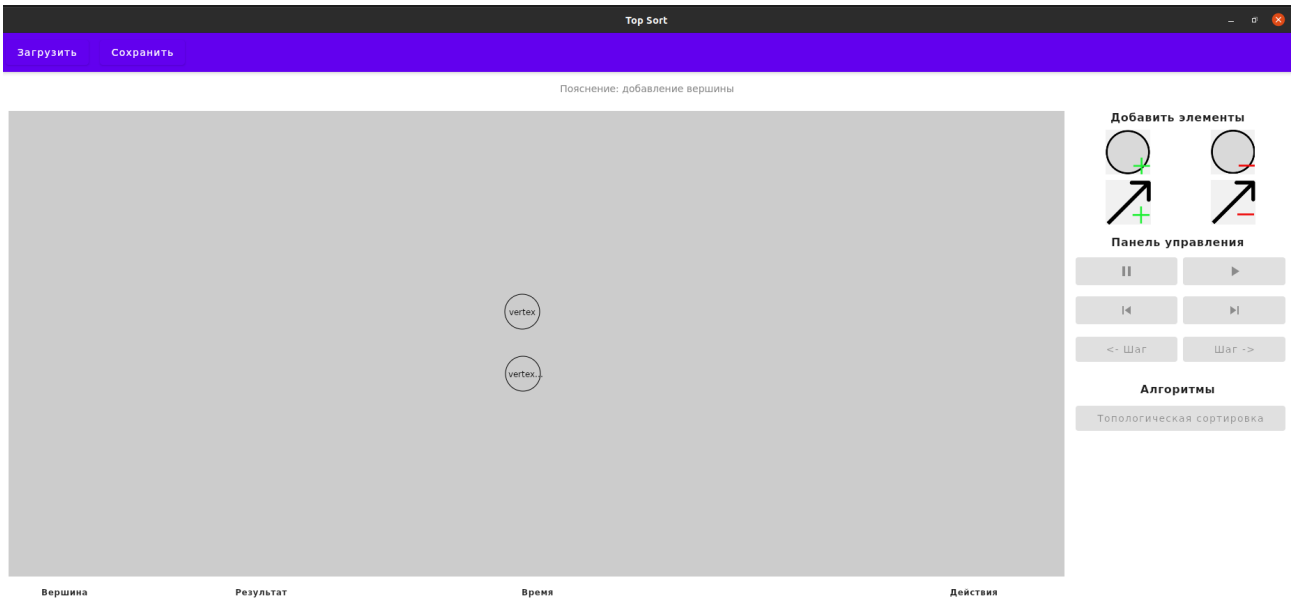

Рис. 13 - Отображение вершины с длинным именем

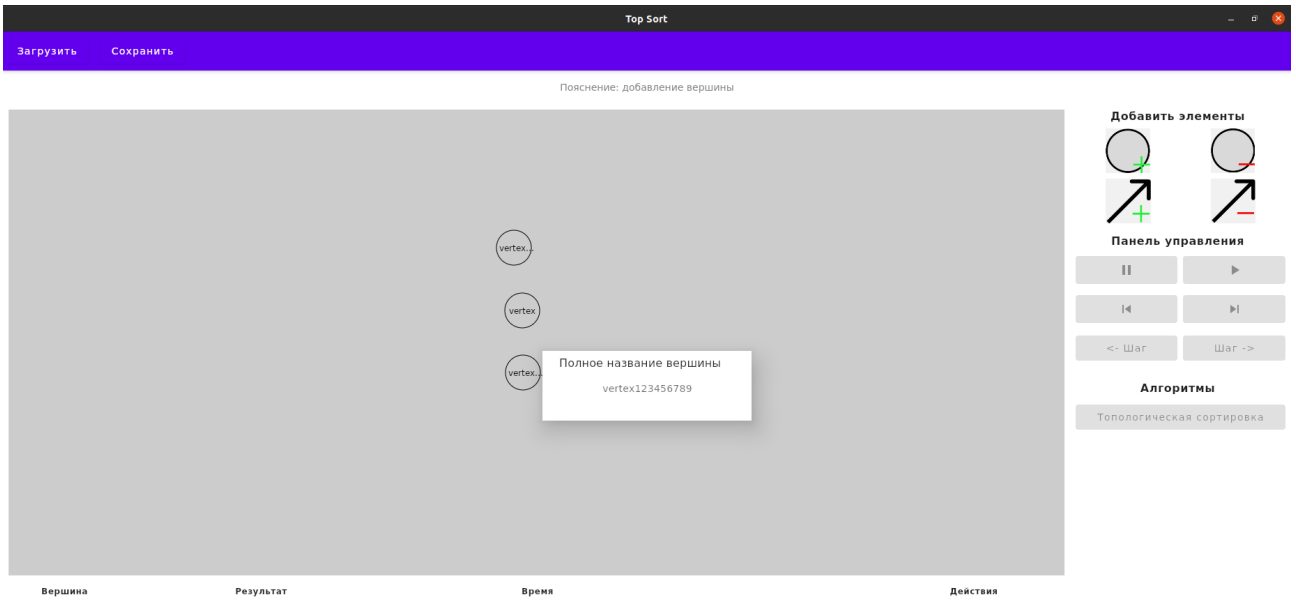

Рис. 14 - Отображение полного имени вершины

4.2. Тестирование кода алгоритма

ЗАКЛЮЧЕНИЕ

При помощи языка программирования Kotlin была написана программа, визуализирующая работу алгоритма топологической сортировки на произвольном графе.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Руководство по языку Kotlin // KotlinLang. URL: https://kotlinlang.ru/ (дата обращения: 28.06.2022).
2. Примеры использования Jetpack Compose для Kotlin Multiplatform // Github. URL: https://github.com/jetbrains/compose-jb (дата обращения: 02.07.2022)
3. Уроки по работе с Jetpack Compose // Jetpack Compose Tutorial. URL: https://www.jetpackcompose.net/jetpack-compose-introduction (дата обращения: 04.07.2022).
4. Руководство по UI-элементам и каталог иконок // Material Design. URL: https://material.io/components (дата обращения: 04.07.2022).
5. Руководство по использованию Coroutines в Kotlin // KotlinLang. URL: https://kotlinlang.org/docs/coroutines-basics.html (дата обращения: 09.07.2022).

Ниже представлены примеры библиографического описания, В КАЧЕСТВЕ НАЗВАНИЯ ИСТОЧНИКА в примерах приводится вариант, в котором применяется то или иное библиографическое описание.

1. Иванов И. И. Книга одного-трех авторов. М.: Издательство, 2010. 000 с.

2. Книга четырех авторов / И. И. Иванов, П. П. Петров, С. С. Сидоров, В. В. Васильев. СПб.: Издательство, 2010. 000 с.

3. Книга пяти и более авторов / И. И. Иванов, П. П. Петров, С. С. Сидоров и др.. СПб.: Издательство, 2010. 000 с.

4. Описание книги под редакцией / под ред. И.И. Иванова СПб., Издательство, 2010. 000 с.

5. Иванов И.И. Описание учебного пособия и текста лекций: учеб. пособие. СПб.: Изд-во СПбГЭТУ «ЛЭТИ», 2010. 000 с.

6. Описание методических указаний / сост.: И.И. Иванов, П.П. Петров. СПб.: Изд-во СПбГЭТУ «ЛЭТИ», 2010. 000 с.

7. Иванов И.И. Описание статьи с одним-тремя авторами из журнала // Название журнала. 2010, вып. (№) 00. С. 000–000.

8. Описание статьи с четырьмя и более авторами из журнала / И. И. Иванов, П. П. Петров, С. С. Сидоров и др. // Название журнала. 2010, вып. (№) 00. С. 000–000.

9. Иванов И.И. Описание тезисов доклада с одним-тремя авторами / Название конференции: тез. докл. III международной науч.-техн. конф., СПб, 00–00 янв. 2000 г. / СПбГЭТУ «ЛЭТИ», СПБ, 2010, С. 000–000.

10. Описание тезисов доклада с четырьмя и более авторами / И. И. Иванов, П. П. Петров, С. С. Сидоров и др. // Название конференции: тез. докл. III международной науч.-техн. конф., СПб, 00–00 янв. 2000 г. / СПбГЭТУ «ЛЭТИ», СПБ, 2010, С. 000–000.

11. Описание электронного ресурса // Наименование сайта. URL: http://east-front.narod.ru/memo/latchford.htm (дата обращения: 00.00.2010).

12. ГОСТ 0.0–00. Описание стандартов. М.: Изд-во стандартов, 2010.

13. Пат. RU 00000000. Описание патентных документов / И. И. Иванов, П. П. Петров, С. С. Сидоров. Опубл. 00.00.2010. Бюл. № 00.

14. Иванов И.И. Описание авторефератов диссертаций: автореф. дисс. канд. техн. наук / СПбГЭТУ «ЛЭТИ», СПБ, 2010.

15. Описание федерального закона: Федер. закон [принят Гос. Думой 00.00.2010] // Собрание законодательств РФ. 2010. № 00. Ст. 00. С. 000–000.

16. Описание федерального постановления: постановление Правительства Рос. Федерации от 00.00.2010 № 00000 // Опубликовавшее издание. 2010. № 0. С. 000–000.

17. Описание указа: указ Президента РФ от 00.00.2010 № 00 // Опубликовавшее издание. 2010. № 0. С. 000–000.

ПРИЛОЖЕНИЕ А
СХЕМА УСТРОЙСТВА АРХИТЕКТУРЫ ПРИЛОЖЕНИЯ

ПРИЛОЖЕНИЕ Б

TOP SORT

полный код программы должен быть в приложении, печатать его не надо