

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

**ОТЧЕТ**  
**по учебной практике**  
**Тема: Сильно связанные компоненты оргграфа**

|                    |       |                |
|--------------------|-------|----------------|
| Студентка гр. 8383 | _____ | Аверина О.С.   |
| Студентка гр. 8383 | _____ | Максимова А.А. |
| Студент гр. 8383   | _____ | Мирсков А.А.   |
| Руководитель       | _____ | Фирсов М.А.    |

Санкт-Петербург  
2020

## ЗАДАНИЕ НА УЧЕБНУЮ ПРАКТИКУ

Студентка Аверина О.С. группы 8383

Студентка Максимова А.А. группы 8383

Студент Мирсков А.А. группы 8383

Тема практики: **Сильно связанные компоненты орграфа**

Задание на практику:

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

Алгоритм: Косарайю.

Сроки прохождения практики: 29.06.2020 – 12.07.2020

Дата сдачи отчета: 00.07.2020

Дата защиты отчета: 00.07.2020

|              |  |                |
|--------------|--|----------------|
| Студентка    |  | Аверина О.С.   |
| Студентка    |  | Максимова А.А. |
| Студент      |  | Мирсков А.А.   |
| Руководитель |  | Фирсов М.А.    |

## **АННОТАЦИЯ**

Целью данной практической работы является изучение и углубление теоретических знаний языка Java, закрепление материала при разработке собственного пошагового визуализатора алгоритма поиска сильно связанных компонент орграфа, обладающего удобным и понятным пользовательским интерфейсом и предусматривающего поведение пользователя, которое без обработки, может приводить к неопределенному поведению программы.

Данная практическая работа состоит из введения, в котором описана спецификация приложения, сопровождаемая макетом меню и диаграммой сценариев, требований, которые должны быть реализованы в программе, прототипа и промежуточных версий кода, плана разработки приложения и распределения ролей в бригаде, описания особенностей реализации, используемых в работе структур данных и разработанных методов, тестирования работы алгоритма и пользовательского интерфейса, а также заключения и списка источников, используемых при написании программы.

## **SUMMARY**

The purpose of this practical work is to study and deepen the theoretical knowledge of the Java language, consolidate the material when developing your own step-by-step visualizer of the search algorithm for strongly connected components of a digraph, which has a convenient and intuitive user interface and provides user behavior that, without processing, can lead to undefined program behavior.

This practical work consists of an introduction, which describes the specification of the application, accompanied by a menu layout and a diagram of scenarios, requirements that must be implemented in the program, prototype and

intermediate versions of the code, an application development plan and role distribution in the team, description of the implementation features used in the work of data structures and developed methods, testing the operation of the algorithm and the user interface, as well as the conclusion and list of sources used when writing the program.

## СОДЕРЖАНИЕ

|      |  |    |
|------|--|----|
|      | Введение   | 6  |
| 1.   | Требования к программе                                 | 7  |
| 1.1. | Исходные требования к программе*                       | 7  |
| 1.2. | Уточнение требований после сдачи прототипа             | 13 |
| 1.3. | Уточнение требований после сдачи 1-ой версии           | 0  |
| 1.4  | Уточнение требований после сдачи 2-ой версии           | 0  |
| 2.   | План разработки и распределение ролей в бригаде        | 14 |
| 2.1. | План разработки  | 14 |
| 2.2. | Распределение ролей в бригаде                          | 15 |
| 3.   | Особенности реализации                                 | 16 |
| 3.1. | Описание алгоритма Косарайю                            | 16 |
| 3.2. | Структуры данных                                       | 0  |
| 3.3  | Основные методы  | 0  |
| 4.   | Тестирование   | 19 |
| 4.1  | План тестирования                                      | 19 |
| 4.2  | Тестирование работы алгоритма и ввода графа из файла.  | 21 |
| 4.3  | ...  | 0  |
|      | Заключение   | 0  |
|      | Список использованных источников                       | 0  |
|      | Приложение А. Исходный код – только в электронном виде | 0  |

## ВВЕДЕНИЕ

Целью выполнения данной практической работы, является разработка программы, на базе высокоуровневого языка Java, реализующей пошаговую визуализацию алгоритма поиска сильно связанных компонент орграфа. Визуализатор алгоритма при этом должен обладать понятным и удобным пользовательским интерфейсом.

Реализация поиска сильно связанных компонент основана на алгоритме Косарайю, в котором ключевым аспектом является поиск в глубину с фиксированием времени выхода из каждой вершины орграфа.

Под сильно связными компонентами орграфа подразумевается его максимальные по включению сильно связанные подграфы. Сильно связанные подграф - это граф, содержащий некое подмножество вершин данного графа и все ребра, инцидентные данному подмножеству, в котором между любыми двумя вершинами, включенными в него, существуют ориентированные пути из  $s$  в  $t$  и из  $t$  в  $s$ , где  $s$  и  $t$  - две любые вершины подграфа.

# 1. ТРЕБОВАНИЯ К ПРОГРАММЕ

## 1.1. Исходные требования к программе

### 1.1.1. Входные данные

Входные данные, а именно орграф, описываемый вершинами и ориентированными невзвешенными ребрами, может быть задан как список ребер, где указывается “Количество ребер”, а затем строки формата “Начальная вершина, конечная вершина” в текстовом файле, импортируемого в программу при нажатии клавиши “Импорт файла”.

Пример:

```
4
1 3
4 2
3 4
3 2
```

Также может быть построен вручную с помощью использования клавиш, располагаемых на панели управления и нажатия на экран с помощью курсора мыши для выбора местоположения вершин или построения ориентированного ребра между ними. Так, чтобы построить граф нужно выполнить следующую последовательность шагов:

1. Нажатие клавиши “Добавление вершины” с последующим щелчком на место, куда будет добавлена вершина
2. Для добавления ориентированного ребра: выбор клавиши “Добавить ребро”, поочередное нажатие на вершины, между которыми будет построено ориентированное ребро (первая вершина начальная, вторая - конечная)
3. Для отмены добавления вершины или ребра необходимо выбрать одну из кнопок “Удаление ребра”, “Удаление вершины” и щелкнуть на объект

### **1.1.2. Визуализация**

Графический интерфейс представляет собой меню, состоящее из панели управления - набора клавиш, предназначенных для вызовов команд, реализующих построение графа, окна, в котором отображается пошаговая визуализация алгоритма или выводится результат работы программы - раскрашенный орграф, в зависимости от выбора пользователя, а также имеется окно, используемое для вывода текстовых пояснений и промежуточной информации.

#### Пошаговая визуализация алгоритма:

- 1) Построение графа, введенного пользователем.
- 2) При визуализации поиска в глубину, вершины и ребра, при посещении будут помечаться цветом. Рядом с вершиной будет написано время выхода.
- 3) Затем запуск поиска компонент сильной связности, окрашивание каждой компоненты в свой цвет.

\*При этом на каждом шаге будет выводиться описание действия.

Эскиз интерфейса, реализующего визуализацию алгоритма поиска сильно связанных компонент орграфа, а также предоставляющего возможность ввода входных данных одним из нескольких предоставленных способов, изображен на рисунке 1.



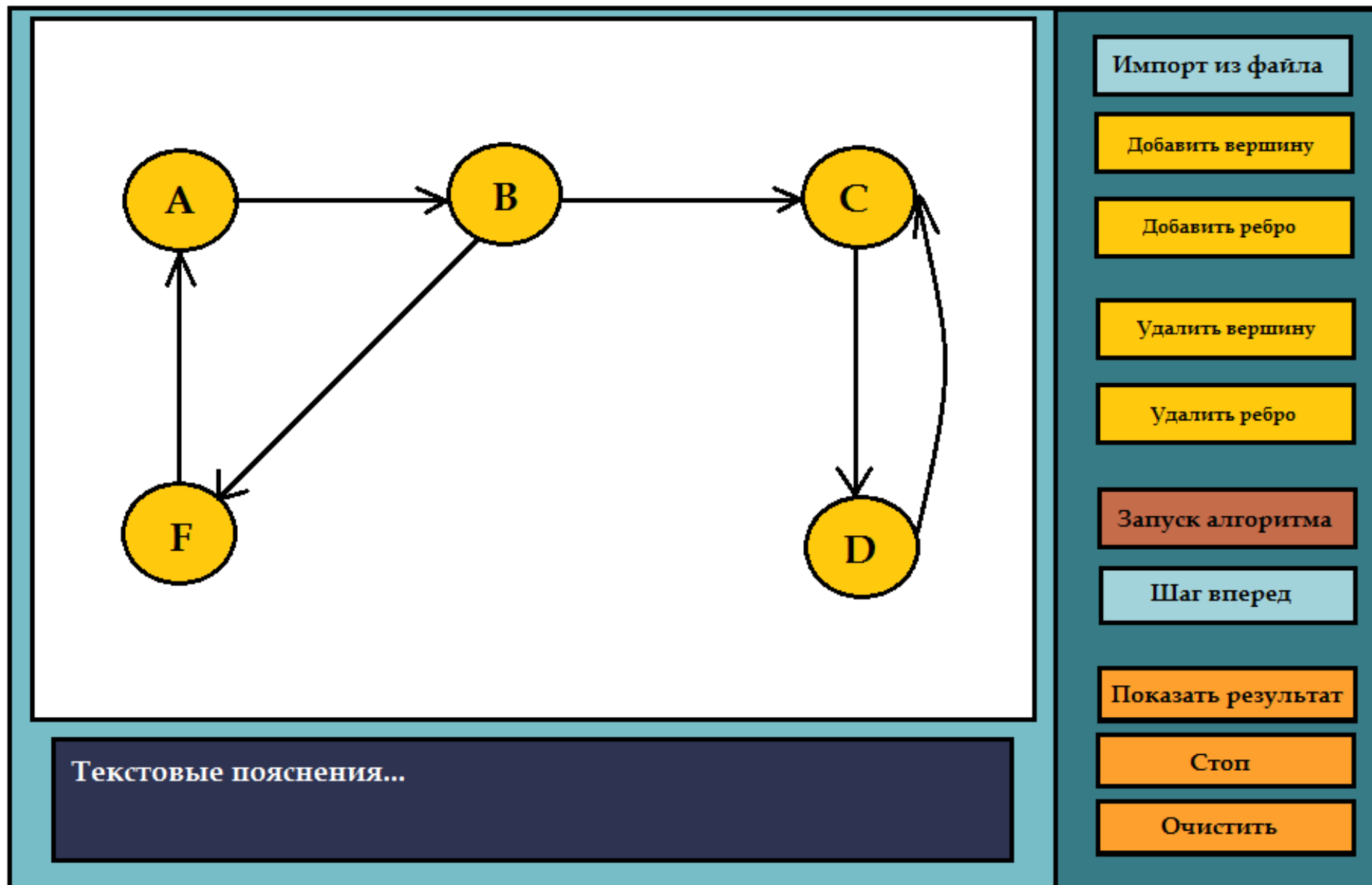


Рис. 1 - Макет меню

### 1.1.3. Взаимодействие пользователя с программой

Весь пользовательский интерфейс описан на панели управления меню - названия клавиш, являются одноименными методам, реализующим интерфейс. Назначение клавиш см. в таблице 1.

| Клавиша              | Назначение  |
|----------------------|---|
| “Импорт из файла”    | Используется для ввода орграфа через файл.  |
| “Добавить вершину”   | Предназначена для добавления новой вершины.   |
| “Добавить ребро”     | Необходима для добавления нового ориентированного ребра.  |
| “Удалить вершину”    | Используется для удаления вершины.  |
| “Удалить ребро”      | Используется для удаления ребра.  |
| “Запуск алгоритма”   | После того, как пользователь ввел граф из файла или построил вручную, становится возможным нажатие данной клавиши, которая запускает работу самого алгоритма, сопровождаемую пошаговой визуализацией. |
| “Шаг вперед”         | Переходит к следующему шагу алгоритма.  |
| “Очистить”           | Очищение экрана. Предоставляет возможность повторного использования программы без перезапуска, начиная с ввода исходного орграфа.   |
| “Показать результат” | Используется для получения графа, после применения к нему алгоритма Косарайю, без пошаговой визуализации.   |
| “Стоп”               | При нажатии на эту клавишу выполнение алгоритма останавливается, введенный граф остается и может быть отредактирован перед повторным запуском алгоритма.  |

Таблица 1 - Клавиши панели управления

Возможные сценарии взаимодействия пользователя с интерфейсом разрабатываемой программы, представлены на рисунке 2.

Во время работы алгоритма в строке состояний снизу от окна графа будут поясняться действия алгоритма, производимые на соответствующем шаге, например: “Найдена компонента связности 1. Запуск поиска в глубину из вершины 5”. По возможности такие сообщения будут подробными, но, не

слишком нагроможденными и будут формулироваться из расчета, что пользователь ранее не был знаком с алгоритмом Косарайю.

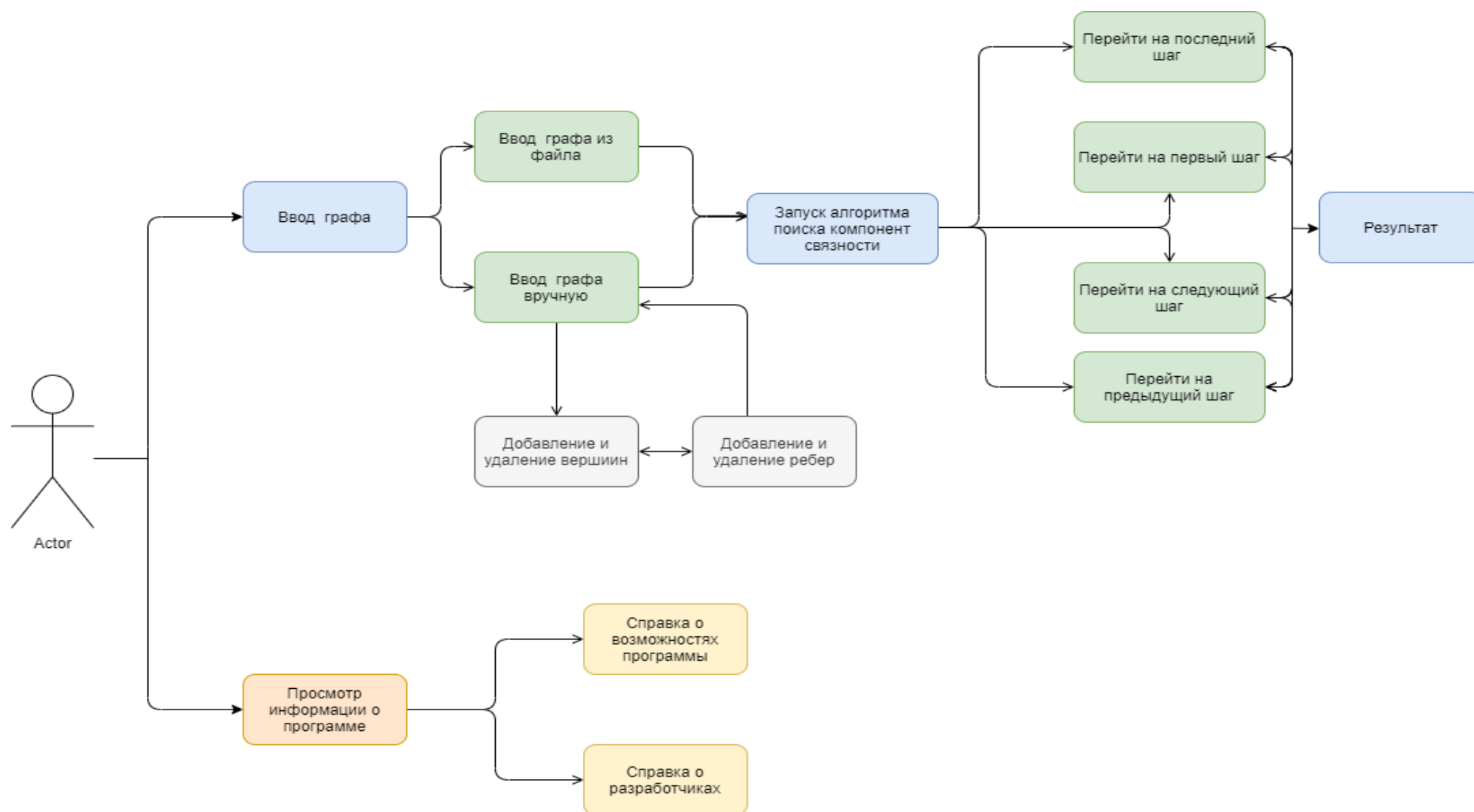


Рис. 2. Диаграмма сценариев

## **1.2. Уточнения требований в ходе разработки**

### **1.2.1. Уточнения требований после сдачи прототипа**

1. Текстовые пояснения должны выделяться и копироваться.
2. При изменении размера окна должен меняться размер области вывода графа.

## 2. ПЛАН РАЗРАБОТКИ И РАСПРЕДЕЛЕНИЕ РОЛЕЙ В БРИГАДЕ

### 2.1. План разработки

Таблица 2 - План разработки программы

| 02.07   | 03.07<br>Показ прототипа   | 04.07   | 05.07  | 06.07<br>Показ 1 версии   |
|---|--|---|--|---|
| + Сдать вводное задание.<br>+ Написать спецификацию.  | + Сдать спецификацию.<br>+ Создать окно интерфейса, добавить кнопки управления.<br>+ Сделать план тестирования<br>+ Сдать прототип программы.  | + Реализовать алгоритм<br>+ Сделать функцию добавления вершины на поле.<br>+ Сделать импорт графа из файла в виде списка ребер и преобразование в список смежности. | + Реализовать добавление ориентированного ребра для двух вершин.<br>+ Создать класс вершины<br>+ Создать класс графа<br>+ Соединить алгоритм и визуализацию. | + Добавить функцию окрашивания вершин, используемую в визуализации<br>+ Подключить клавиши: “Импорт из файла”, “Добавление ребра”, “Добавление вершины”, “Запуск”.<br>+ Провести тестирования<br>+ Исправить недочеты |
| 07.07   | 08.07<br>Показ 2 версии  | 09.07   | 10.07<br>Показ финальной версии  | 11.07<br>Резервный день   |
| + Добавить пометки времени выхода вершин при обходе в глубину.<br>+ Добавить удаление ребра и вершины.<br>+ Добавить функцию очистки поля и подключить кнопку “Очистить”. | + Реализовать пошаговую визуализацию алгоритма.<br>+ Подключить клавишу “ Шаг вперед”, “Показать результат”.<br>+ Провести тестирования.<br>+ Исправить недочеты недочеты.<br>+ Сделать текстовые пояснения. | + Подключить клавишу “Стоп”.<br>+ Провести финальное тестирование.<br>+ Исправить недочеты.<br>+ Доделать отчет.  | + Сдать финальную версию.  |   |

## 2.2. Распределение ролей в бригаде

Распределение ролей в процессе разработки представлено в таблице 3.

Таблица 3 - Распределение ролей

| Роль         | Ответственный(е)                   |
|--------------|------------------------------------|
| Лидер        | Ольга Аверина                      |
| Алгоритмист  | Анастасия Максимова                |
| Фронтенд     | Андрей Мирсков, Ольга Аверина      |
| Тестировщик  | Ольга Аверина, Анастасия Максимова |
| Документация | Анастасия Максимова                |

Пояснение:

- **Лидер** - имеет решающее право голоса, занимается управлением репозитория, помогает фронтенду
- **Алгоритмист** - реализует основной алгоритм программы (поиска компонент сильной связности), отвечает за его работу и обеспечивает его безопасность от пользователя (обрабатывает ошибки, которые возможно предвидеть заранее)
- **Фронтенд** - реализация пользовательского интерфейса, отвечает за его наполнение, реализацию, включая пошаговую визуализацию алгоритма
- **Тестировщик** - организует тестирование функционала, создает наборы тестовых данных, знает, что тестировалось, а что нет и по какой причине
- **Документация** - создание документации проекта в виде отчета, выбор формата комментариев, используемых в программе, следит за их написанием

### 3. ОСОБЕННОСТИ РЕАЛИЗАЦИИ

Для разработки графического интерфейса используется библиотека JavaFX.

#### 3.1. Описание алгоритма Косарайю.

Для реализации алгоритма, выполняющего поиск сильно связанных компонент в ориентированном графе, были написаны три класса: `Event`, `Vertex`, `AlgorithmKosarayu`, описанные в разделах 3.2. Структуры данных и 3.3. Основные методы.

На вход алгоритм принимает список ребер, а на выходе возвращает список событий, используемый для пошаговой реализации алгоритма.

Список событий имеет следующий вид: `ArrayList <Event> events`, где `Event` - класс, описывающий в конкретный момент времени одно из девяти событий:

**Событие 1:** Запуск метода `dfs1`, выполняющего поиск в глубину, от вершины  $V_1$ .

**Событие 2:** Конец `dfs1` - у текущей вершины  $V_1$  не осталось соседних необработанных вершин и все рекурсивные вызовы метода уже выполнены.

**Событие 3:** Переход по ребру (во время выполнения `dfs1`) в графе из вершины  $V_1$  в вершину  $V_2$ .

**Событие 4:** У текущей вершины (во время выполнения `dfs1`)  $V_1$  не остались необработанные соседние вершины. Возврат на вершину  $V_2$ , из которой попали в  $V_1$ .

**Событие 5:** Инвертирование графа - изменение направления ребер на противоположные, за исключением петель.

**Событие 6:** Запуск метода `dfs2`, выполняющего поиск в глубину, от вершины  $V_1$ , которая принадлежит компоненте  $X$ .

**Событие 7:** Конец `dfs2` - у текущей вершины  $V_1$  не осталось необработанных соседних вершин и все рекурсивные вызовы метода уже выполнены.



**Событие 8:** Переход по ребру (во время выполнения dfs2) в графе из вершины  $V_1$  в вершину  $V_2$ , принадлежащих компоненте  $X$ .

**Событие 9:** У текущей вершины (во время выполнения dfs2)  $V_1$  не остались необработанные соседние вершины. Возврат на вершину  $V_2$ , из которой попали в  $V_1$ .

**Событие 10:** Сигнализирует, о следующей, по порядку времени выхода, вершине.

Алгоритм можно представить в виде следующей последовательности шагов:

**Шаг 1.** Создание графа (массив вершин): список ребер преобразуется в списки смежности вершин.

**Шаг 2.** Запуск поиска в глубину (на исходном графе), из необработанных вершин, с фиксированием времени выхода из них (dfs1):

**Шаг 2.1.** Помечаем вершину, как обрабатываемую "0".

**Шаг 2.2.** Запоминаем время входа в вершину.

**Шаг 2.3.** Ищем смежные необработанные вершины, если нет, переход на Шаг 2.4.

**Шаг 2.3.1.** Если нашли, проверяем: петля?

**Шаг 2.3.1. 1.** Да, пропускаем вершину и возвращаемся к шагу 2.3.

**Шаг 2.3.1. 2.** Нет - переход на шаг 2.3.2.

**Шаг 2.3.2.** Выполняем рекурсивный вызов поиска в глубину от найденной вершины, запоминая, из какой вершины в нее попали.

**Шаг 2.4.** Помечаем вершину, как обработанную "1".

**Шаг 2.5.** Запоминаем время выхода из вершины.

**Шаг 3.** Инвертируем дуги исходного ориентированного графа:

**Шаг 3.1.** В исходном списке ребер меняем вершины местами, кроме петель.

**Шаг 3.2.** Используем преобразованный список, для инициализации инвертированного графа.

**Шаг 4.** Запуск поиска в глубину (на инвертированном графе), из необработанных вершин, выбирая вершины в порядке уменьшения времени выхода из них, полученном на шаге 2:

**Шаг 4.1.** Помечаем вершину, как обрабатываемую "0".

**Шаг 4.2.** Ищем смежные необработанные вершины, если нет, переход на шаг 4.3.

**Шаг 4.2.1.** Если нашли, проверяем: петля?

**Шаг 4.2.1.1.** Да, пропускаем вершину и возвращаемся к шагу 4.2.

**Шаг 4.2.1.2.** Нет - переход на шаг 4.2.2.

**Шаг 4.2.2.** Фиксируем принадлежность к компоненте сильной связности.

**Шаг 4.2.3.** Выполняем рекурсивный вызов поиска в глубину от найденной вершины, запоминая, из какой вершины в нее попали.

**Шаг 4.3.** Помечаем вершину, как обработанную "1".

**Шаг 5.** Снова инвертируем граф - получаем исходный.

Примечание: вершины, имеющие одинаковые значения поля `component` образуют деревья, являющиеся сильно связными компонентами исходного графа.

## **3.2. Структуры данных**

## **3.3. Основные методы**

## 4. ТЕСТИРОВАНИЕ

### 4.1. План тестирования

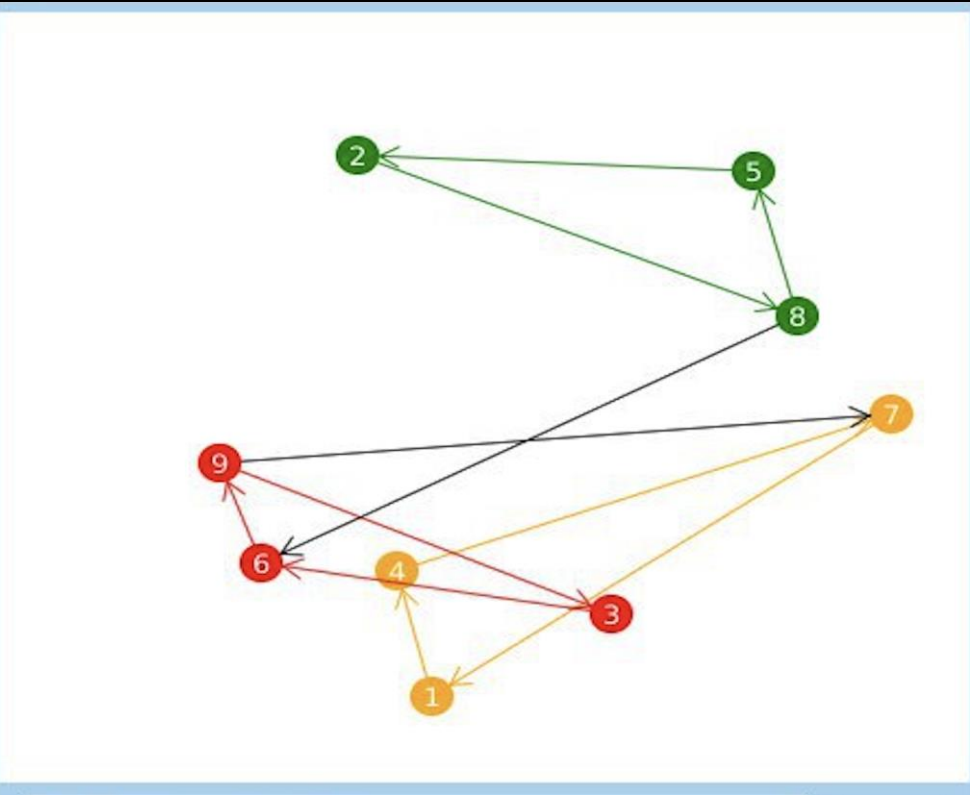
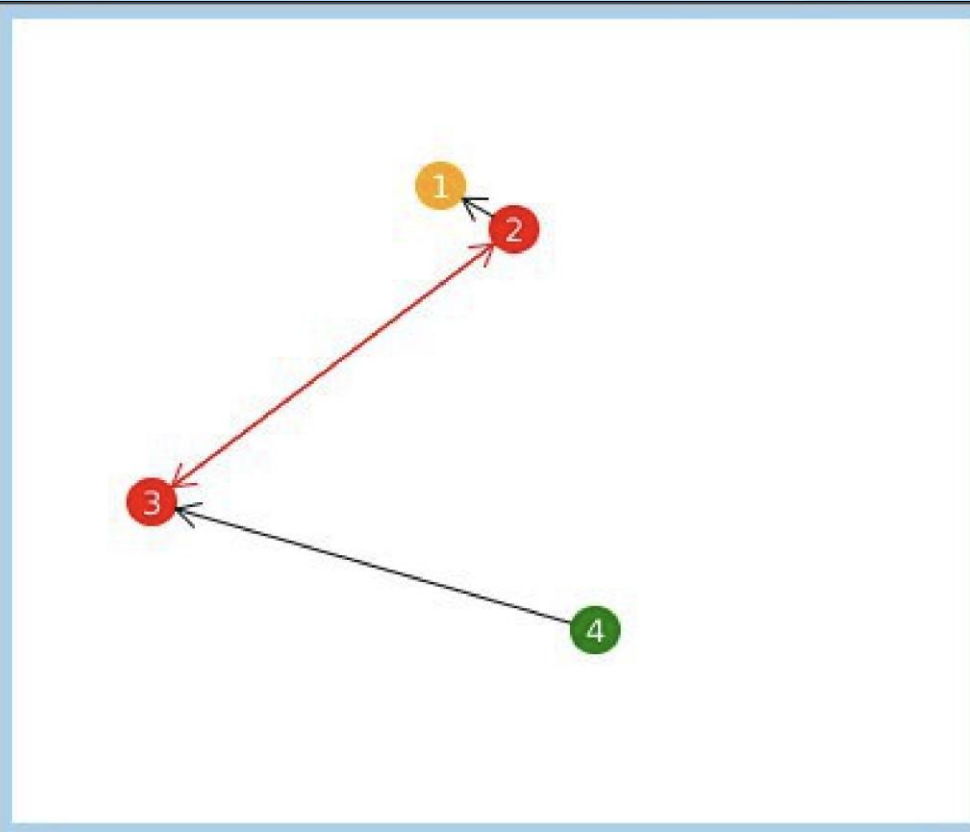
1. В процессе разработки предстоит протестировать следующее:
  - a. Ручной ввод графа.
  - b. Ввод графа из файла.
  - c. Корректность работы алгоритма
  - d. Пошаговая работа алгоритма и раскраска графа.
  - e. Работа кнопок.
2. Для этого будут использоваться следующие методы тестирования:
  - a. Ручной ввод графа.
    - i. Добавление вершин, ребер
    - ii. Удаление вершин, ребер
    - iii.
  - b. Ввод графа из файла.
    - i. Ввод в файл некорректных данных отслеживание реакции программы. Некорректные данные — это, например, ввод из файла с одним числом в файле, или когда в строке с “ребром” введен один номер вершины.
  - c. Корректность работы алгоритма
    - i. Проверка работы алгоритма на графе из одной вершины, из двух несмежных вершин.
    - ii. Проверка работы на входных данных и проверка корректности результата
  - d. Пошаговая работа алгоритма и раскраска графа.
    - i. Запуск и пошаговый “прогон” алгоритма с отслеживанием шагов и окраски соответствующих вершин.
  - e. Работа кнопок.
    - i. Попытка нажать на кнопки “Удалить вершины/ребра” при отсутствии на поле вершин/ребер, соответственно.

- ii. Попытка во время работы алгоритма нажать кнопки “Импорт из файла”, “Добавить вершины/ребра”, “Удалить вершины/ребра”, “Запуск”.

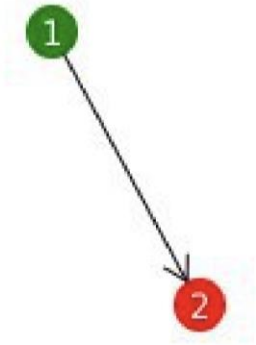
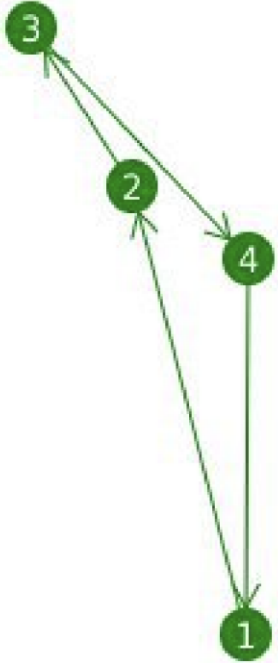
3. Тестирование проводит на следующих этапах разработки:

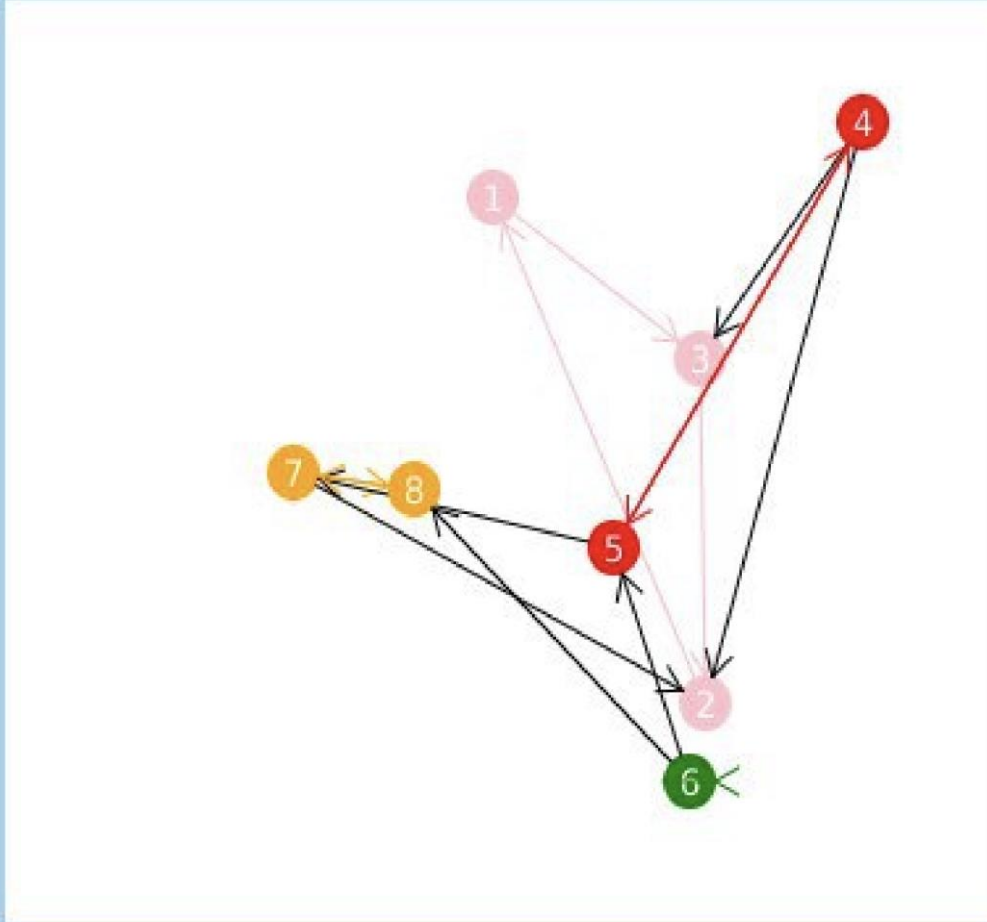
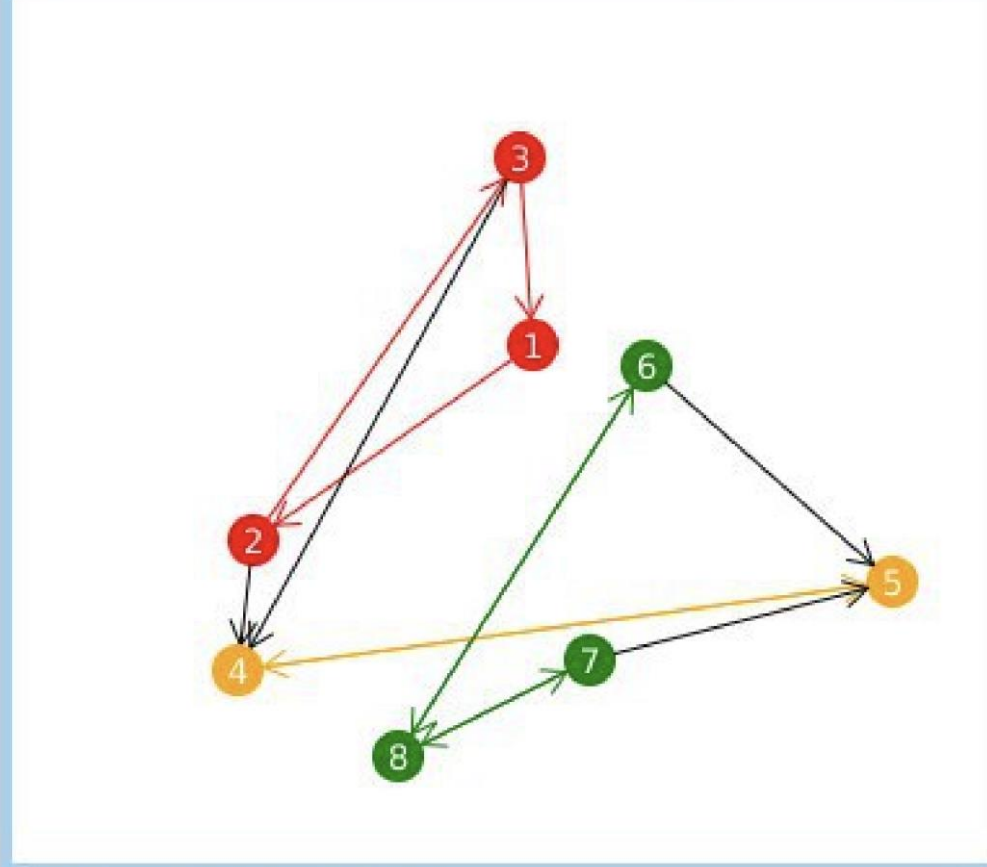
- a. Ручной ввод графа: между сдачей 1 и 2 версий программы.
- b. Ввод графа из файла: перед сдачей 1 версии.
- c. Корректность работы алгоритма: перед сдачей 1 версии.
- d. Пошаговая работа алгоритма и раскраска графа: перед сдачей 2 версии.
- e. Работа кнопок: перед сдачей финальной версии.

#### 4.2. Тестирование работы алгоритма и ввода графа из файла.


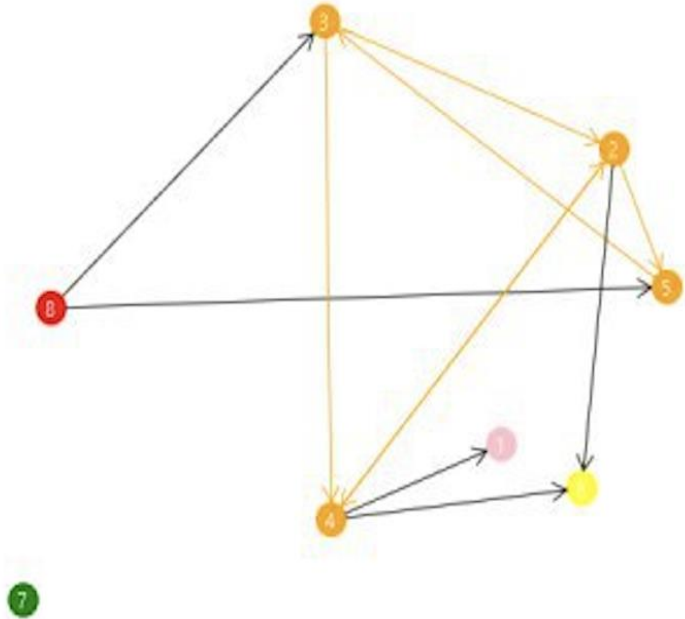
| Входные данные   | Выходные данные  |
|--|--|
| <p>7 1</p> <p>4 7</p> <p>1 4</p> <p>9 7</p> <p>6 9</p> <p>3 6</p> <p>9 3</p> <p>8 6</p> <p>2 8</p> <p>5 2</p> <p>8 5</p> |   |
| <p>2 1</p> <p>3 2</p> <p>2 3</p> <p>4 3</p>  |  |

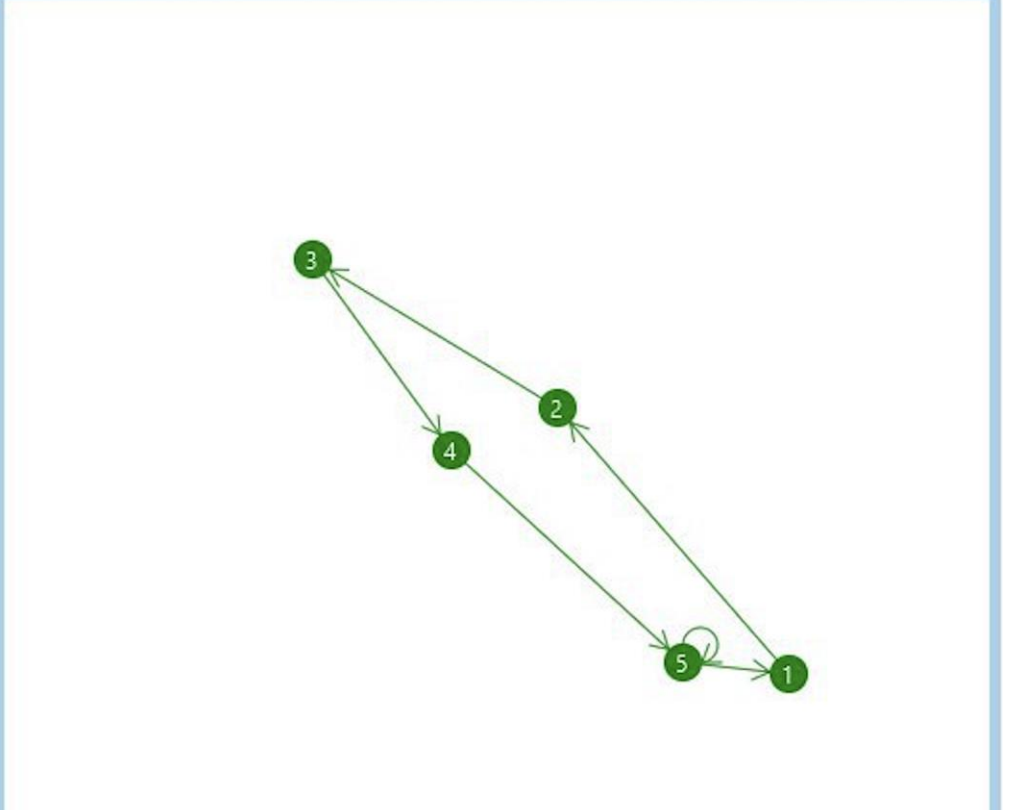
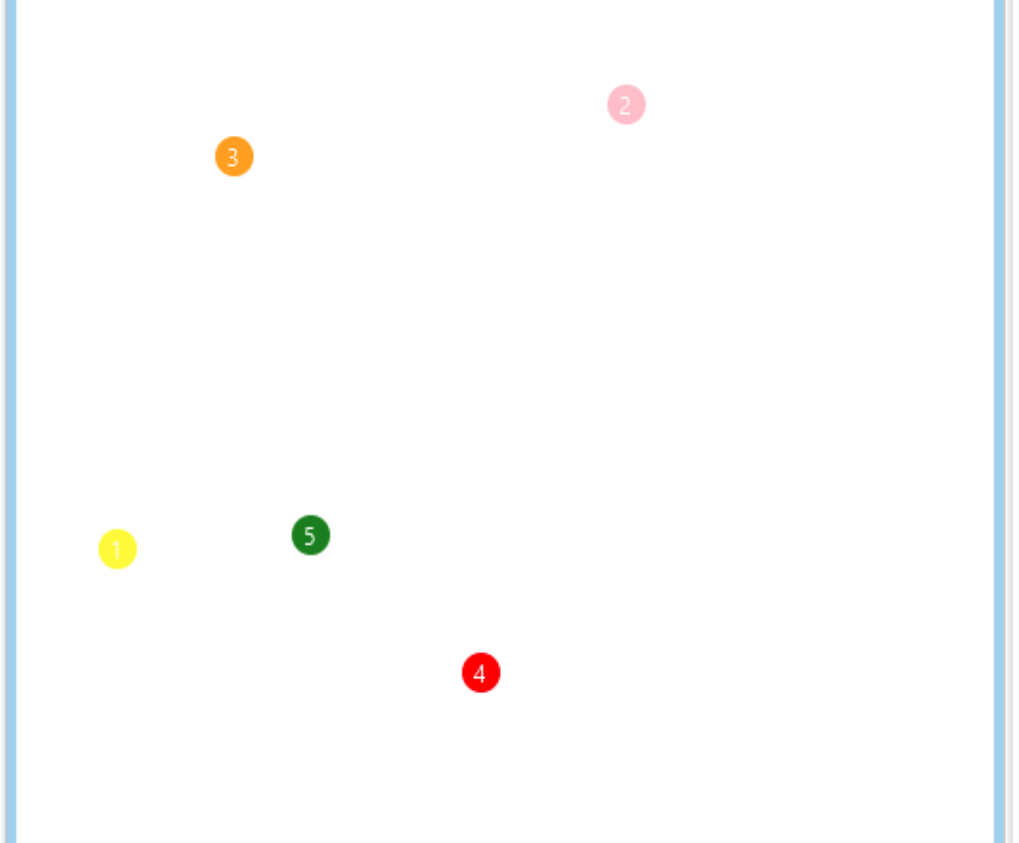
|  |   |
|--|---|
| <div data-bbox="325 338 376 701"> <p>1 3</p> <p>5 1</p> <p>3 5</p> <p>3 4</p> <p>4 2</p> <p>2 4</p> </div>   | <div data-bbox="826 271 1398 846"> <pre> graph TD     1((1)) -- green --&gt; 3((3))     3 -- green --&gt; 4((4))     4 -- green --&gt; 3     4 -- red --&gt; 2((2))     3 -- green --&gt; 5((5))     5 -- green --&gt; 1 </pre> </div>  |
| <div data-bbox="325 1133 376 1749"> <p>2 1</p> <p>1 2</p> <p>3 1</p> <p>3 2</p> <p>4 2</p> <p>3 5</p> <p>5 4</p> <p>4 5</p> <p>4 6</p> <p>5 6</p> </div> | <div data-bbox="687 1039 1369 1742"> <pre> graph TD     1((1)) -- pink --&gt; 2((2))     2 -- pink --&gt; 3((3))     3 -- green --&gt; 5((5))     5 -- green --&gt; 4((4))     4 -- pink --&gt; 2     3 -- green --&gt; 6((6))     5 -- green --&gt; 6     6 -- orange --&gt; 4 </pre> </div> |

|                                    |  |
|------------------------------------|--|
| <p>1 2</p>                         |  <p>A directed graph with two nodes. Node 1 is a green circle at the top left, and node 2 is a red circle at the bottom right. A black arrow points from node 1 to node 2.</p>   |
| <p>1 2<br/>4 1<br/>3 4<br/>2 3</p> |  <p>A directed graph with four nodes, all green circles. Node 3 is at the top left, node 2 is below it, node 4 is to the right of node 2, and node 1 is at the bottom right. The edges are: 3 to 2, 3 to 4, 2 to 1, and 4 to 1.</p> |


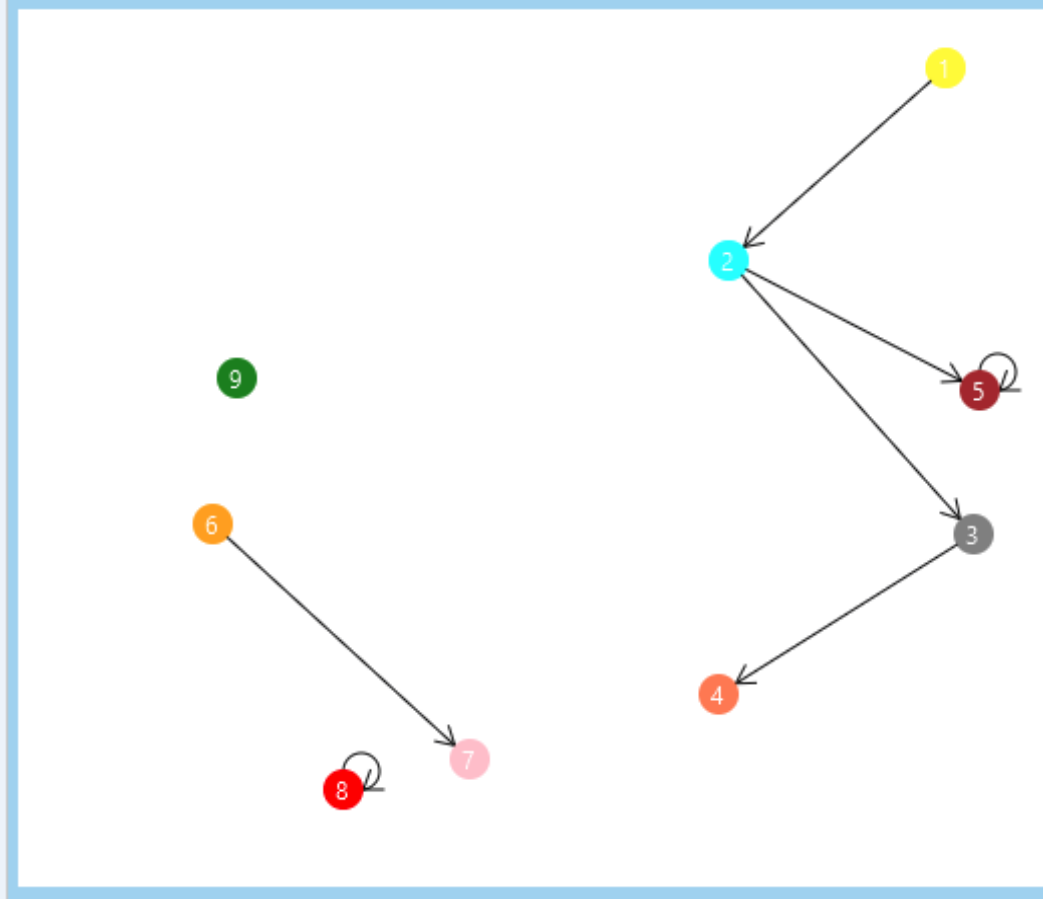
|  |  |
|--|--|
| <div data-bbox="239 78 462 1019"> <div>1 3</div> <div>3 2</div> <div>2 1</div> <div>4 3</div> <div>4 2</div> <div>7 2</div> <div>4 5</div> <div>5 4</div> <div>5 7</div> <div>7 8</div> <div>8 7</div> <div>6 5</div> <div>6 6</div> <div>6 8</div> </div> | <div data-bbox="478 78 1500 1019">  </div>     |
| <div data-bbox="239 1030 462 1912"> <div>1 2</div> <div>3 1</div> <div>2 3</div> <div>2 4</div> <div>3 4</div> <div>4 5</div> <div>5 4</div> <div>6 5</div> <div>7 5</div> <div>8 6</div> <div>6 8</div> <div>8 7</div> <div>7 8</div> </div>              | <div data-bbox="478 1030 1500 1912">  </div> |



|   |  |
|---|--|
| 1 -1  |     |
| 7 -1<br>8 5<br>8 3<br>2 6<br>2 5<br>2 4<br>3 2<br>3 4<br>4 2<br>4 1<br>4 6<br>5 3 |  |

|   |   |
|---|---|
| <p>1 2</p> <p>2 3</p> <p>3 4</p> <p>4 5</p> <p>5 1</p> <p>5 5</p> |    |
| <p>1 -1</p> <p>2 -1</p> <p>3 -1</p> <p>4 -1</p> <p>5 -1</p>       |  |

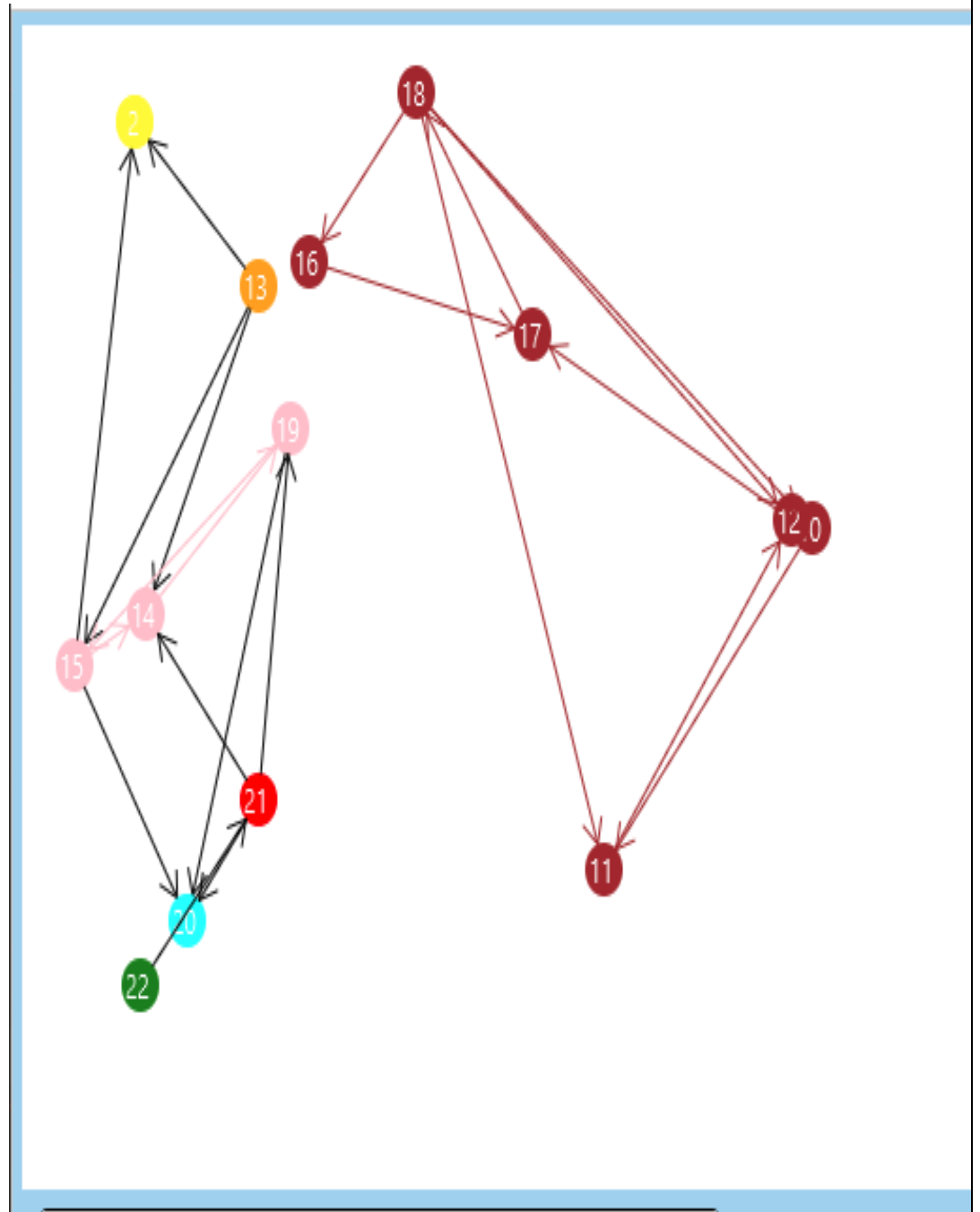
|   |  |
|---|--|
| <div data-bbox="300 212 352 1093" data-label="Text"> <p> 1 3<br/> 3 2<br/> 2 1<br/> 4 3<br/> 4 2<br/> 7 2<br/> 4 5<br/> 5 4<br/> 5 7<br/> 7 8<br/> 8 7<br/> 6 5<br/> 6 6<br/> 6 8 </p> </div> | <div data-bbox="459 85 1516 1227" data-label="Diagram"> </div>   |
| <div data-bbox="300 1534 352 1646" data-label="Text"> <p> 1 -1<br/> 1 1 </p> </div>   | <div data-bbox="587 1254 1388 1926" data-label="Diagram"> </div> |

|  |  |
|--|--|
| <p>1 1</p> <p>2 2</p> <p>3 3</p> <p>4 4</p>  |     |
| <p>1 2</p> <p>2 3</p> <p>2 5</p> <p>5 5</p> <p>3 4</p> <p>6 7</p> <p>8 8</p> <p>9 -1</p> |  |

|  |   |
|--|---|
| <div data-bbox="271 123 359 952"> <p>1 2</p> <p>2 3</p> <p>2 4</p> <p>1 3</p> <p>1 4</p> <p>3 5</p> <p>5 5</p> <p>2 6</p> <p>6 7</p> <p>8 -1</p> <p>9 10</p> <p>10 11</p> <p>11 9</p> </div> | <div data-bbox="446 73 1516 985"> </div>    |
| <div data-bbox="271 1086 359 1848"> <p>1 2</p> <p>2 3</p> <p>3 4</p> <p>2 5</p> <p>5 6</p> <p>5 7</p> <p>9 10</p> <p>10 11</p> <p>10 10</p> <p>10 12</p> <p>12 13</p> <p>13 13</p> </div>    | <div data-bbox="446 1008 1516 1926"> </div> |

|   |   |
|---|---|
| <div data-bbox="284 159 368 904"> <p>1 2</p> <p>2 3</p> <p>3 4</p> <p>4 5</p> <p>4 6</p> <p>3 7</p> <p>7 8</p> <p>7 9</p> <p>10 7</p> <p>9 9</p> <p>11 -1</p> <p>12 13</p> </div> | <div data-bbox="464 91 1493 981"> </div>    |
| <div data-bbox="293 1099 352 1845"> <p>1 2</p> <p>2 6</p> <p>2 4</p> <p>3 5</p> <p>3 4</p> <p>4 7</p> <p>4 2</p> <p>5 9</p> <p>6 4</p> <p>6 7</p> <p>7 1</p> <p>9 8</p> </div>    | <div data-bbox="464 1014 1493 1933"> </div> |

10 11  
11 12  
12 17  
13 14  
13 15  
13 2  
14 19  
15 2  
15 14  
15 20  
16 17  
17 18  
18 12  
18 16  
18 11  
18 10  
19 20  
19 15  
21 20  
21 19  
21 14  
22 21



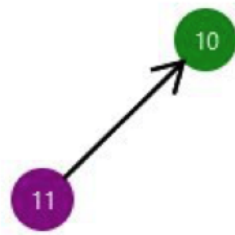
1 2

1 2

2 3

3 4

4 1





## **ЗАКЛЮЧЕНИЕ**

Кратко подвести итоги, проанализировать соответствие поставленной цели и полученного результата.

## СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

*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.

**ПРИЛОЖЕНИЕ А**  
**НАЗВАНИЕ ПРИЛОЖЕНИЯ**

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