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

ОТЧЕТ

по учебной практике

Тема: разработка приложения, визуализирующего работу алгоритма поиска мостов в графе на языке Java.

Студент гр. 0383 _____ Подопригора И.П.

Студент гр. 0383 _____ Пенкин М.В.

Студент гр. 0383 _____ Позолотин К.С.

Руководитель _____ Ефремов М.А.

Санкт-Петербург

2022

ЗАДАНИЕ
НА УЧЕБНУЮ ПРАКТИКУ

Студент Подопригора И.П. группы 0383

Студент Пенкин М.В. группы 0383

Студент Позолотин К.С. группы 0383

Тема практики: разработка приложения, визуализирующего работу алгоритма поиска мостов в графе на языке Java.

Задание на практику:

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

Алгоритм: поиск мостов в графе.

Сроки прохождения практики: 29.06.2020 – 12.07.2020

Дата сдачи отчета: 12.07.2020

Дата защиты отчета: 12.07.2020

Студент гр. 0383	_____	Подопригора И.П.
Студент гр. 0383	_____	Пенкин М.В.
Студент гр. 0383	_____	Позолотин К.С.
Руководитель	_____	Ефремов М.А.

АННОТАЦИЯ

Цель данной практики заключается в изучении языка программирования Java и применении полученных знаний путём создания приложения с графическим интерфейсом, реализующего визуализацию алгоритма поиска мостов в графе и удобное взаимодействие с пользователем.

SUMMARY

The purpose of this practice is to learn the Java programming language and apply the knowledge gained by creating an application with a graphical interface that implements the visualization of the algorithm for finding bridges in a graph and convenient user interaction.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Цель практики: разработать визуализатор алгоритма поиска мостов в графе на языке Java с графическим интерфейсом.

Задачи практики:

1. Изучить новый язык программирования Java и его основные средства.
2. Научиться разработке в команде с использованием системы контроля версий Git.
3. Реализовать выбранный алгоритм на языке Java с визуализацией и графическим интерфейсом.
4. Защитить разработанный проект.

Реализуемый алгоритм:

Алгоритм поиска мостов в графе. Пусть дан неориентированный граф. Мостом называется такое ребро, удаление которого делает граф несвязным (или, точнее, увеличивает число компонент связности). Требуется найти все мосты в заданном графе.

В начале работы алгоритма запустим обход в глубину из произвольной вершины графа; обозначим её через *root*. Заметим следующий факт: пусть мы находимся в обходе в глубину, просматривая сейчас все рёбра из вершины *v*. Тогда, если текущее ребро *(v, to)* таково, что из вершины *to* и из любого её потомка в дереве обхода в глубину нет обратного ребра в вершину *v* или какого-либо её предка, то это ребро является мостом. В противном случае оно мостом не является. Теперь осталось научиться проверять этот факт для каждой вершины эффективно. Для этого воспользуемся "временами входа в вершину", вычисляемыми алгоритмом поиска в глубину.

Итак, пусть *t_in[v]* — это время захода поиска в глубину в вершину v. Теперь введём массив *f_up[v]*, который и позволит нам отвечать на вышеописанные запросы. Время *f_up[v]* равно минимуму из времени захода в саму вершину *t_in[v]*, времён захода в каждую вершину p, являющуюся концом некоторого обратного ребра *(v, p)*, а также из всех значений *f_up[to]* для каждой

вершины to, являющейся непосредственным сыном v в дереве поиска. Тогда, из вершины v или её потомка есть обратное ребро в её предка тогда и только тогда, когда найдётся такой сын *to*, что $f\_up[to] \geq t\_in[v]$. Таким образом, если для текущего ребра *(v, to)* (принадлежащего дереву поиска) выполняется $f\_up[to] > t\_in[v]$, то это ребро является мостом; в противном случае оно мостом не является.

1. ТРЕБОВАНИЯ К ПРОГРАММЕ

1.1. Исходные Требования к программе

Программа должна содержать графический интерфейс, понятный для пользователя.

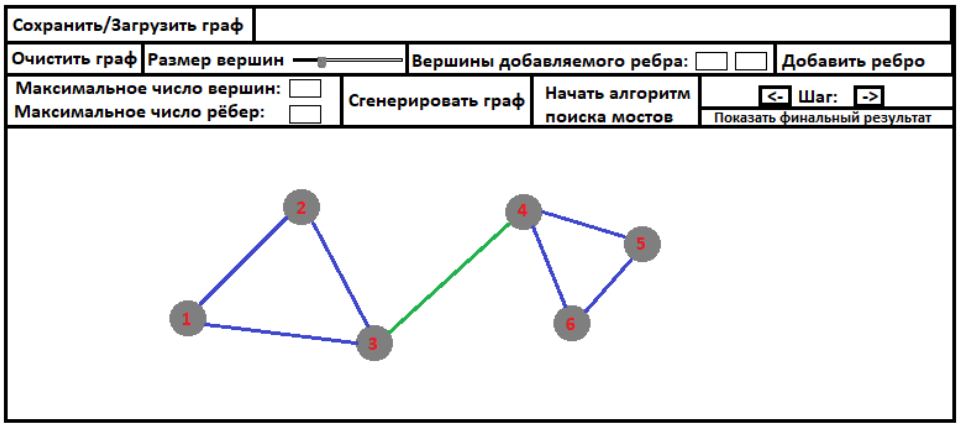

Рис. 1. Эскиз графического интерфейса

Должна быть возможность задавать входные данные через графический интерфейс: добавление вершин в текущий граф по щелчку мыши, и добавление ребра между указанными вершинами по кнопке. Также должно быть реализовано задание графа через файл и с помощью случайной генерации.

Результат работы алгоритма на текущем заданном графе должен отображаться по нажатию соответствующей кнопки: рёбра-мосты должны стать обозначенными отличным от изначального цветом.

1.2. Диаграмма сценариев использования

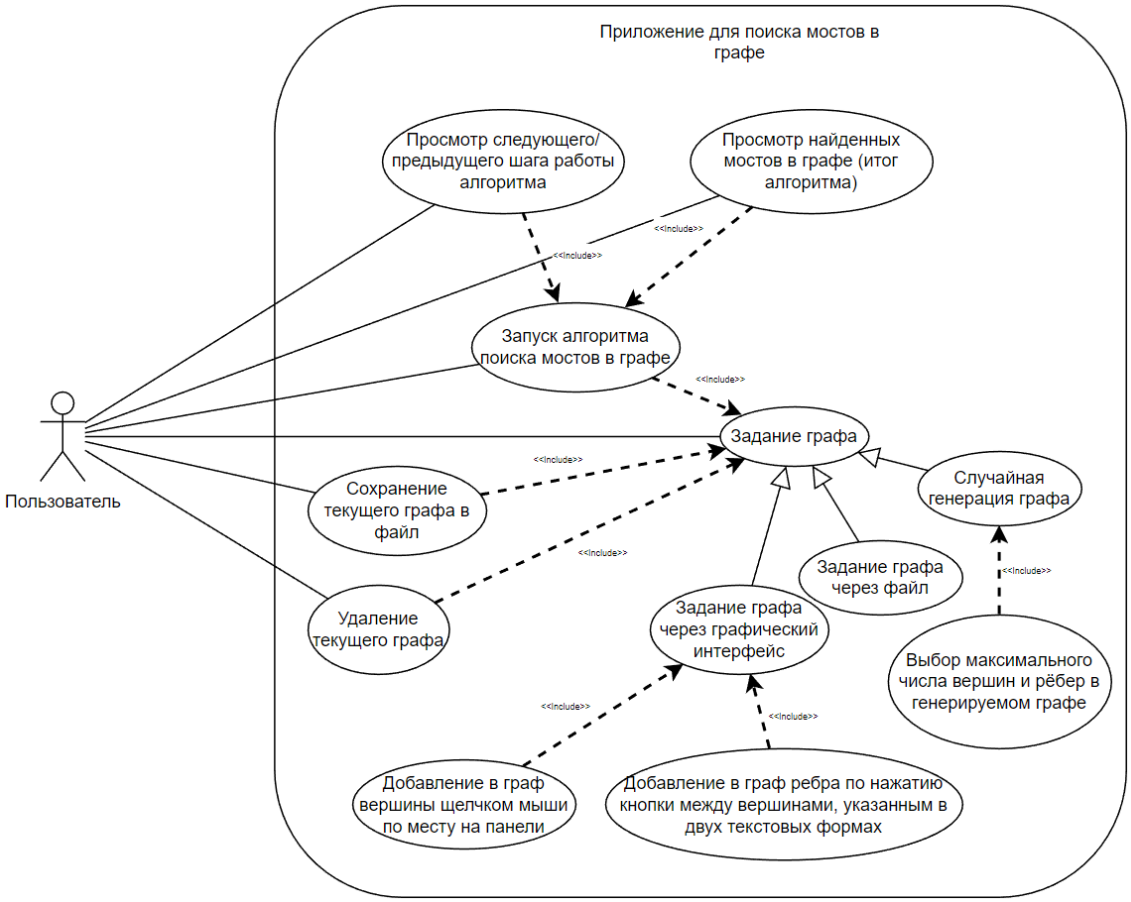

Рис. 2. Диаграмма сценариев использования.

2. ПЛАН РАЗРАБОТКИ И РАСПРЕДЕЛЕНИЕ РОЛЕЙ В БРИГАДЕ

2.1. План разработки

1. Согласование спецификации и плана разработки – **2 июля**;
2. Разработка прототипа приложения с реализованным (возможно неполным) графическим интерфейсом, хотя бы одним способом задать входные данные и с реализованной возможностью увидеть результат работы алгоритма - до **4 июля**;
3. Утверждение первой версии приложения - **4 июля**
4. Создание финальной версии приложения - до **8 июля**.
5. Утверждение финальной версии приложения - **8 июля**.

2.2. Распределение ролей в бригаде

Подопригора И.П. - реализация основных элементов графического интерфейса, его связи с реализуемым алгоритмом и реализация задания входных данных через графический интерфейс.

Пенкин М.В. - реализация алгоритма поиска мостов в графе и его тестирование.

Позолотин К.С. - реализация считывания графа из файла, случайной генерации графа.

3. РЕАЛИЗАЦИЯ ПРИЛОЖЕНИЯ

3.1. Интерфейс приложения.

Реализованный прототип графического интерфейса приложения представлен на рис. 3.

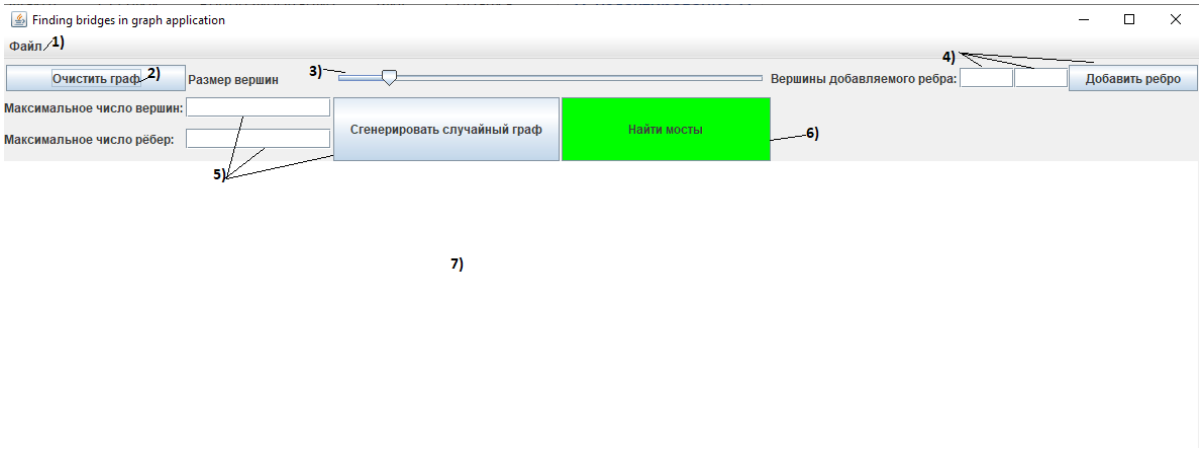

Рис. 3. Элементы графического интерфейса.

1) - Меню работы с файлами, содержащее элементы, отвечающие за загрузку графа из файла и сохранение графа в файл.
2) Кнопка очистки текущего графа (удаление всех вершин и рёбер).
3) Ползунок, отвечающий за размер отображаемых вершин графа.
4) Элементы, отвечающие за добавление ребра в граф: кнопка добавления ребра и две текстовые формы, в которые заносятся номера рёбер, которые необходимо соединить ребром.
5) Элементы, отвечающие за случайную генерацию графа: кнопка генерации графа и две текстовые формы, в которые заносятся максимальное количество вершин в генерируемом графе и максимальное число рёбер.
6) Кнопка, отвечающая за пометку рёбер, являющихся мостами в текущем графе.
7) Область, в которой задаётся граф.

3.2. Реализованные классы и методы.

Класс Graph содержит информацию о графе в виде списка смежности, вершины в графе имеют номера от 0 до n-1 (n - число вершин). Граф имеет поле для хранения информации о рёбрах - мостах. В методе clear() граф очищается, в методах addVertex() и addEdge(int v1, int v2) происходит добавление вершины и ребра в граф соответственно.

Класс DrawableGraph наследуется от Graph и дополнительно содержит информацию о координатах вершин.

Класс GraphDrawPanel реализует панель, на которой отрисовывается граф. В методе paintComponent(Graphics g) происходит отрисовка графа на панели.

Класс AppFrame содержит элементы графического интерфейса и поле с объектом класса DrawableGraph.

Класс RandomGraphCreator генерирует случайный граф в методе CreateRandomGraph, принимающем в аргументах ссылку нас объект графа и информацию об ограничениях на максимальное количество вершин и рёбер, а также границы координат, по которым вершины будут располагаться на поле.

Класс BridgesFinder реализует алгоритм поиска мостов в графе в методе findBridges, принимающем в аргументах объект графа и возвращающем список ребер-мостов в графе.

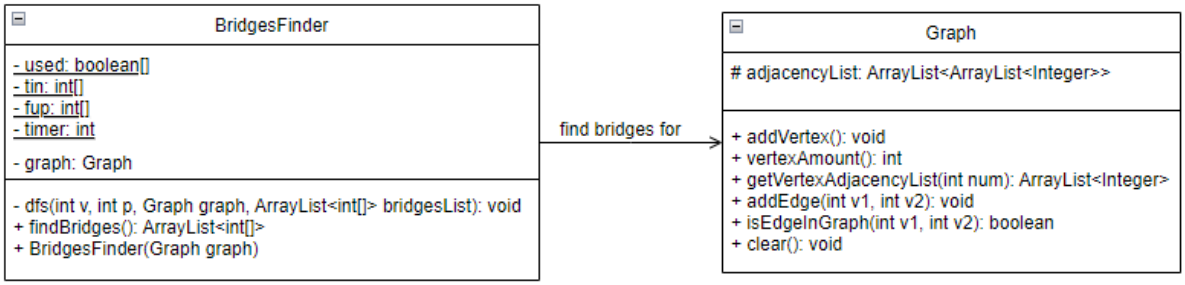

Рис. 4. UML-диаграмма классов, отвечающих за хранение графа и реализацию алгоритма.

4. ТЕСТИРОВАНИЕ

4.1. Первый подраздел третьего раздела

4.2. Второй подраздел третьего раздела

ЗАКЛЮЧЕНИЕ

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

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

ПРИЛОЖЕНИЕ А
ИСХОДНЫЙ КОД ПРИЛОЖЕНИЯ

https://github.com/PodoprigoraIvan/Bridges_in_graph_Java