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

ОТЧЕТ

по учебной практике

Тема: Визуализация алгоритмов на языке Kotlin

Студент гр. 8381 _____ Сосновский Д.Н.

Студент гр. 8381 _____ Киреев К.А.

Студент гр. 8381 _____ Муковский Д.В.

Руководитель _____ Фирсов М.А.

Санкт-Петербург
2020

ЗАДАНИЕ

НА УЧЕБНУЮ ПРАКТИКУ

Студент Сосновский Д.Н. группы 8381

Студент Киреев К.А. группы 8381

Студент Муковский Д.В. группы 8381

Тема практики: визуализация алгоритмов на языке Kotlin

Задание на практику:

Командная итеративная разработка визуализатора алгоритма на Kotlin с графическим интерфейсом.

Алгоритм: Дейкстры.

Сроки прохождения практики: 29.06.2020 – 12.07.2020

Дата сдачи отчета: ??.07.2020

Дата защиты отчета: ??.07.2020

Студент	_____	Сосновский Д.Н.
Студент	_____	Киреев К.А.
Студент	_____	Муковский Д.В.
Руководитель	_____	Фирсов М.А.

ВВЕДЕНИЕ

Цели выполнения учебной практики:

1. Освоение среды программирования Kotlin, особенностей и синтаксиса данного языка.

2. Изучение средств для реализации графического интерфейса, а именно – библиотеки JavaFX.

3. Получение таких навыков как:

- разработка программ на языке Kotlin;
- работа с системой контроля версий, репозиториями;
- командная работа.

Поставленные цели будут достигнуты во время выполнения проекта (визуализации алгоритма Дейкстры), выбранного бригадой.

Алгоритм Дейкстры - алгоритм на графах, изобретённый нидерландским учёным Эдсгером Дейкстрой в 1959 году. Данный алгоритм находит кратчайшие пути от одной из вершин графа до всех остальных вершин. Алгоритм работает только для графов без рёбер отрицательного веса. Алгоритм широко применяется в программировании и технологиях, например, при планировании автомобильных и авиамаршрутов, при разводке электронных плат, в протоколах маршрутизации.

1. ПОСТАНОВКА ЗАДАЧИ

1.1. Формулировка задания

Разработать программу, которая визуализирует алгоритм Дейкстры для поиска кратчайших путей в графе на языке программирования Kotlin.

1.2. Формальное определение и сложность алгоритма

Дан взвешенный ориентированный граф G(V,E) без дуг отрицательного веса. Найти кратчайшие пути от некоторой вершины a графа G до всех остальных вершин этого графа.

Сложность алгоритма Дейкстры зависит от способа нахождения вершины v, а также способа хранения множества непосещённых вершин и способа обновления меток. В простейшем случае, когда для поиска вершины с минимальным d[v] просматривается всё множество вершин, а для хранения величин d используется массив, время работы алгоритма есть O(n^2).

1.3. Реализуемый алгоритм

Каждой вершине из V сопоставим метку — минимальное известное расстояние от этой вершины до a. Алгоритм работает пошагово — на каждом шаге он «посещает» одну вершину и пытается уменьшать метки. Работа алгоритма завершается, когда все вершины посещены.

Инициализация.

Метка самой вершины a полагается равной 0, метки остальных вершин — бесконечности. Это отражает то, что расстояния от a до других вершин пока неизвестны. Все вершины графа помечаются как непосещённые.

Шаг алгоритма.

Если все вершины посещены, алгоритм завершается.

В противном случае, из ещё не посещённых вершин выбирается вершина u, имеющая минимальную метку.

Мы рассматриваем всевозможные маршруты, в которых u является предпоследним пунктом. Вершины, в которые ведут рёбра из u, назовём

соседями этой вершины. Для каждого соседа вершины u, кроме отмеченных как посещённые, рассмотрим новую длину пути, равную сумме значений текущей метки u и длины ребра, соединяющего u с этим соседом. Если полученное значение длины меньше значения метки соседа, заменим значение метки полученным значением длины. Рассмотрев всех соседей, пометим вершину u как посещённую и повторим шаг алгоритма.

1.4. Псевдокод

```
func dijkstra(s):
    for v ∈ V
        d[v] = ∞
        used[v] = false
    d[s] = 0
    for i ∈ V
        v = null
        for j ∈ V                          // найдём вершину с минимальным расстоянием
            if !used[j] and (v == null or d[j] < d[v])
                v = j
        if d[v] == ∞
            break
        used[v] = true
        for e : исходящие из v рёбра      // произведём релаксацию по всем рёбрам, исходящим из v
            if d[v] + e.len < d[e.to]
                d[e.to] = d[v] + e.len
```

2. СПЕЦИФИКАЦИЯ

2.1.1. Описание интерфейса

Интерфейс может меняться в ходе разработки. Первоначальный макет представлен на рис. 1

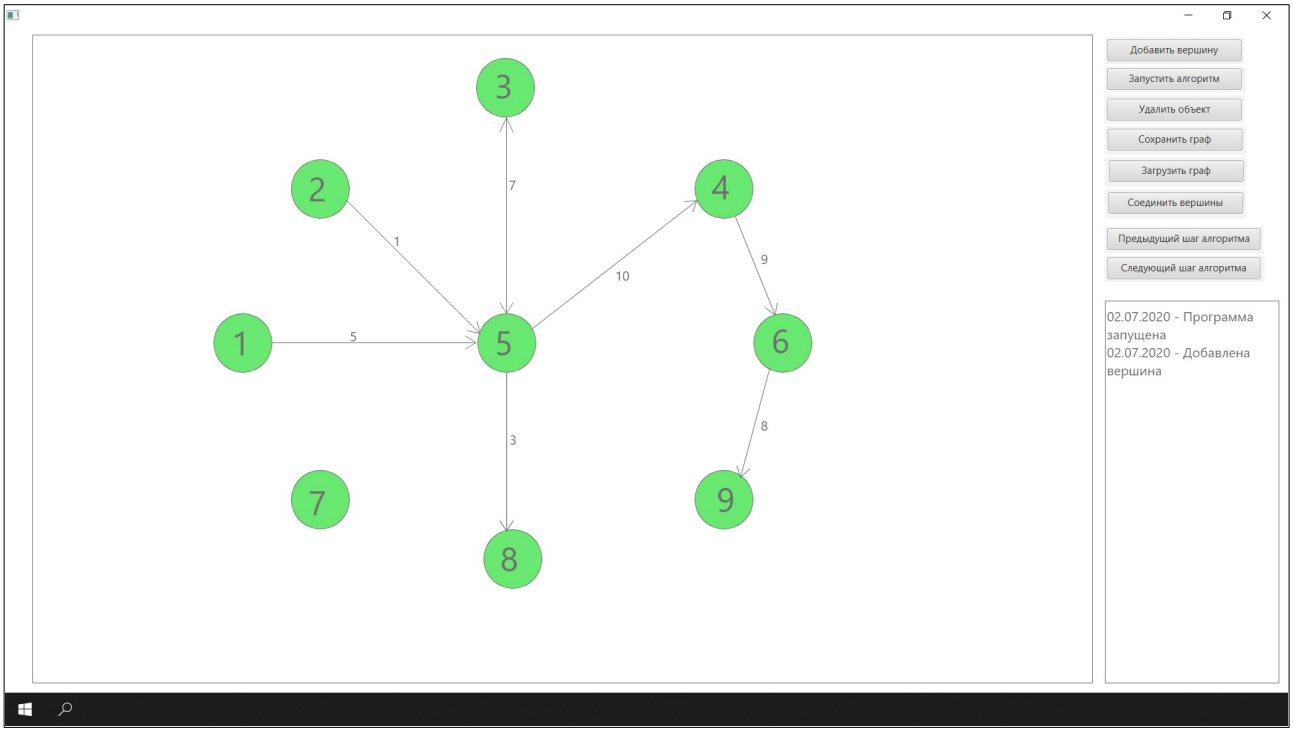

Рисунок 1 — Макет программы

2.1.2. Требования к вводу исходных данных

На вход алгоритму должен подаваться взвешенный ориентированный граф. Именем вершины могут быть цифры. Область создание графа предоставляет возможность ввести данные вручную, с файла или сгенерировать. При ручном вводе достаточно просто вписать данные в необходимое поле. При считывании с файла надо нажать соответствующую кнопку. При генерации нужно ввести необходимые данные в поля генерации.

2.1.3. Требования к визуализации

Пользовательский интерфейс должен представлять собой диалоговое окно, содержащее набор кнопок, предназначенных для управления состоянием программы.

Диалоговое окно должно состоять из:

- Рабочей области для построения графа.
- Кнопки «Сохранить», которая должна позволять пользователю сохранить созданный в рабочей области граф в виде .txt файла.
- Кнопки «Загрузить», которая должна позволять пользователю загрузить граф, для которого необходимо применить алгоритм Дейкстры, в виде .txt файла.
- Кнопки «Добавить вершину», которая должна создать вершину графа в рабочей области.
- Кнопки «Соединить вершины», которая должна создать направленное ребро между двумя вершинами и задать его вес.
- Кнопки «Удалить», которая должна удалить выбранный пользователем элемент графа.
- Кнопки «Вперед», которая должна отобразить следующую итерацию алгоритма.
- Кнопки «Назад», которая должна отобразить предыдущую итерацию алгоритма

2.2. Диаграммы

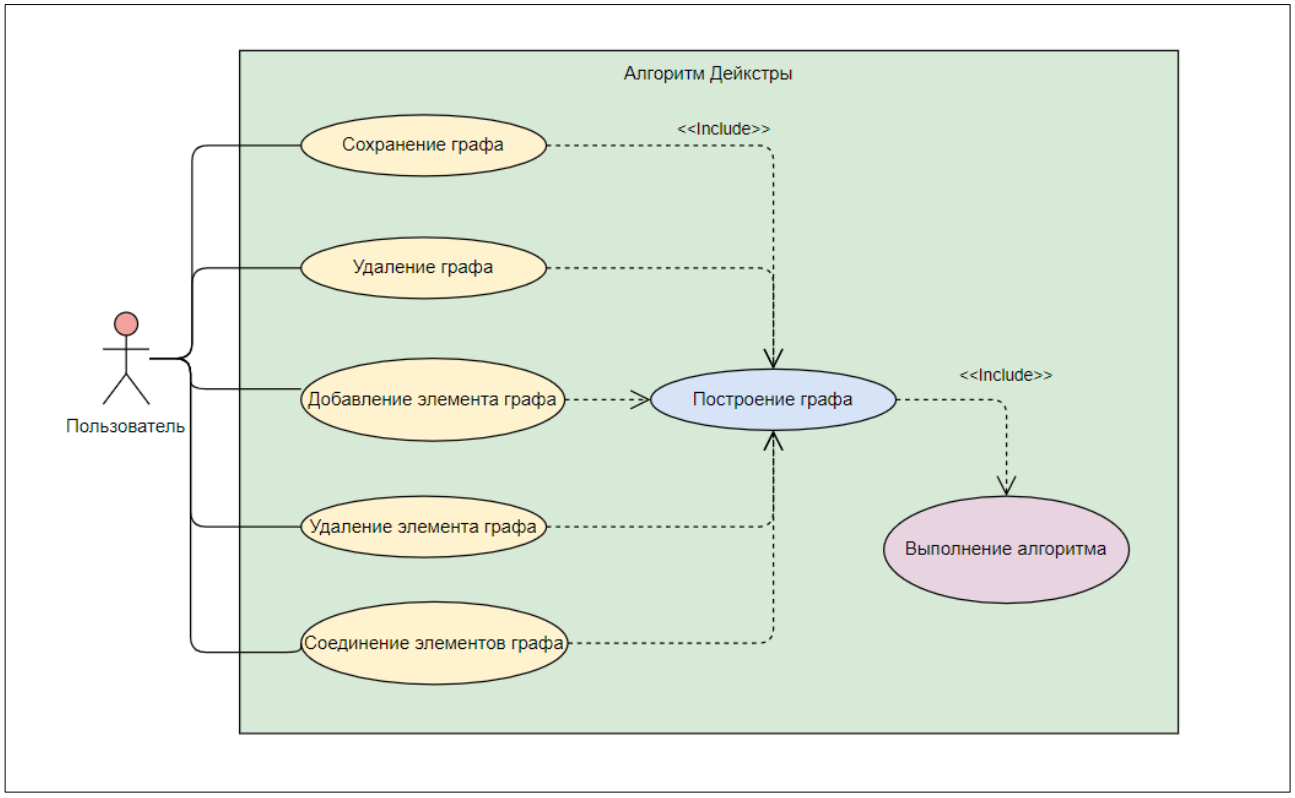

Рисунок 2 - Use case диаграмма

3. ПЛАН РАЗРАБОТКИ И РАСПРЕДЕЛЕНИЕ РОЛЕЙ В БРИГАДЕ

3.1. План разработки

Разработка программы будет вестись с использованием языка программирования Kotlin и его возможностей по созданию GUI. Для написания программы используется интегрированная среда разработки IntelliJ IDEA.

Репозиторий с исходным кодом и данной спецификацией расположен на GitHub по адресу: https://github.com/DmitriySosnovskiy/kotlin_training_practice

План разработки программы разбит на следующие шаги:

1. Создание прототипа (03.07): к этому шагу будет создано приложение, демонстрирующее интерфейс, но почти не реализующее основные функции;

2. Первая версия (06.07): на этом шаге будет реализован класс, отвечающий за работу самого алгоритма Дейкстры. Затем он будет соединен с классом GUI, используя интерфейсы обоих классов. В результате получится начальная рабочая версия программы;

3. Вторая версия (08.07): реализация пошаговой работы программы;

4. Третья (конечная) версия (10.07): добавление возможности генерации входных данных. Исправление ошибок проекта. Написание тестов к проекту.

3.2. Распределение ролей в бригаде

Роли в бригаде распределены следующим образом:

- Сосновский Дмитрий: реализация GUI и методов взаимодействия с ним;
- Киреев Константин: реализации алгоритма Дейкстры и методов взаимодействия с ним; написание отчета.
- Муковский Даниил: реализация архитектуры программы; проектирование системы классов и их взаимодействия;