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

ОТЧЁТ

по учебной практике

Тема: Визуализация алгоритма Форда-Беллмана.

Студент гр. 9383 _____ Камзолов Н.А.

Студент гр. 9383 _____ Гладких А.А.

Студент гр. 9383 _____ Моисейченко К.А.

Руководитель _____ Фиалковский М.С.

Санкт-Петербург

2021

ЗАДАНИЕ

НА УЧЕБНУЮ ПРАКТИКУ

Студент Камзолов Н.А. группы 9383

Студент Гладких А.А. группы 9383

Студент Моисейченко К.А. группы 9383

Тема практики: Визуализация алгоритма Форда-Беллмана

Задание на практику:

Командная итеративная разработка визуализатора алгоритма на Kotlin с графическим интерфейсом.

Алгоритм: алгоритм Форда-Беллмана

Сроки прохождения практики: 1.07.2021 – 14.07.2021

Дата сдачи отчета: 14.07.2021

Дата защиты отчета: 14.07.2021

Студент гр. 9383		Камзолов Н.А.
Студент гр. 9383		Гладких А.А.
Студент гр. 9383		Моисейченко К.А.
Руководитель		Фиалковский М.С.

АННОТАЦИЯ

Основная цель практики – изучение основ языка программирования Kotlin и разработка приложения с графическим интерфейсом на Kotlin (визуализатора алгоритма Форда-Беллмана). В процессе работы предстоит реализовать алгоритм Форда-Беллмана, разработать прототип интерфейса приложения, протестировать написанную программу и исправить найденные в ней ошибки. Нашей командой было решено разработать визуализацию алгоритма в качестве Android-приложения.

SUMMARY

The main goal of the practice is to learn basics of Kotlin programming language and develop an application with a graphical interface in Kotlin (Ford-Bellman algorithm visualizer). In the work we will have to implement the Ford-Bellman algorithm, develop a prototype of the application interface, test the program and fix the errors if found. Our team decided to develop a visualization of the algorithm as an Android application.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Задача практики состоит в разработке Android-приложения, визуализирующего работу алгоритма Форда-Беллмана. Граф задаётся пользователем. Алгоритм находит кратчайшие пути от одной из вершин графа до всех остальных. Граф и начальная вершина задаются пользователем.

1. ТРЕБОВАНИЯ К ПРОГРАММЕ

1.1. Описание задачи

Приложение будет показывать работу алгоритма Форда-Беллмана по шагам. В приложении можно будет редактировать граф (добавлять, удалять вершины и рёбра) и выбирать начальную вершину для поиска кратчайшего пути.

1.2. Архитектура проекта

Основной архитектурный паттерн - MVVM (Model-View-ViewModel). Таким образом для каждого экрана приложения создан:

1. Layout файл, который отвечает за графическое отображение всех элементов UI(View).
2. Класс ViewModel, который отвечает за всю логику экрана и хранение данных на данный момент времени (ViewModel).
3. Класс Fragment, который отвечает за обновление UI, является посредником между View и ViewModel.

1.3. Формат входных и выходных данных

В качестве входных данных принимается заданный пользователем в режиме редактирования граф, а также начальная вершина для поиска кратчайших путей.

1.4. Основные типы данных

1. Список смежности для работы с графом в алгоритме. Представляет из себя map<String, Neighbours>, где Neighbours - массив пар из имени соседней вершины и длине ребра, проведенного к этой вершине.

2. Список смежности для корректного отображения графа. Представляет из себя map<String, VertexInfo>, но где VertexInfo – это большая `структура данных для хранения информации об отображении вершины и всех

инцидентных ей ребер. При вводе графа пользователем, вся информация об отображении графа попадает именно сюда.

1.5. Интерфейс

Интерфейс программы состоит из:

Начальное окно с кнопками начала работы с алгоритмом («Опробовать алгоритм»), подробного пояснения алгоритма («Как работает?»), список разработчиков («Разработчики») и кнопкой выхода из приложения.

Рисунок 1 – Прототип интерфейса начального окна

Рисунок 2 – Прототип окна пояснения алгоритма

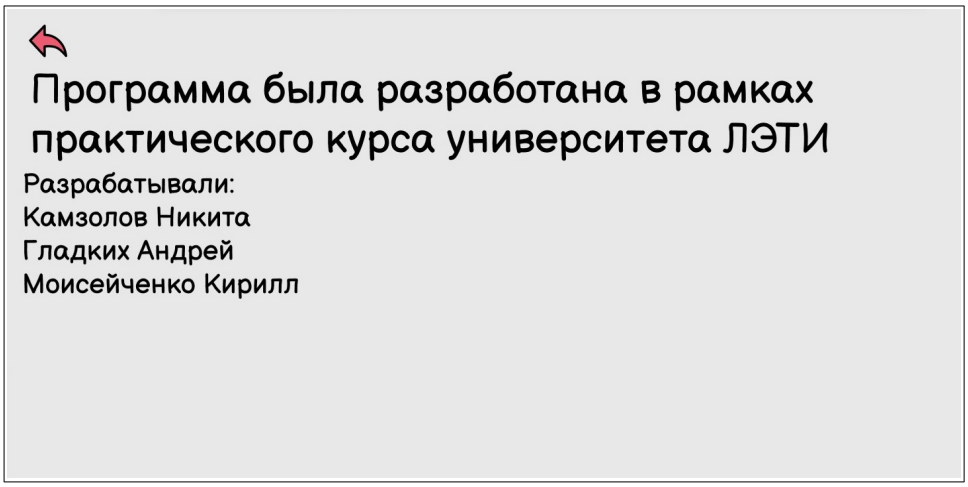

Рисунок 3 – Прототип окна «Разработчики»

Интерфейс основного окна состоит из кнопок «Назад», «Шаги алгоритма», «Помощь», «Редактировать граф», «Удалить вершину», «Удалить ребро», «Завершить редактирование».

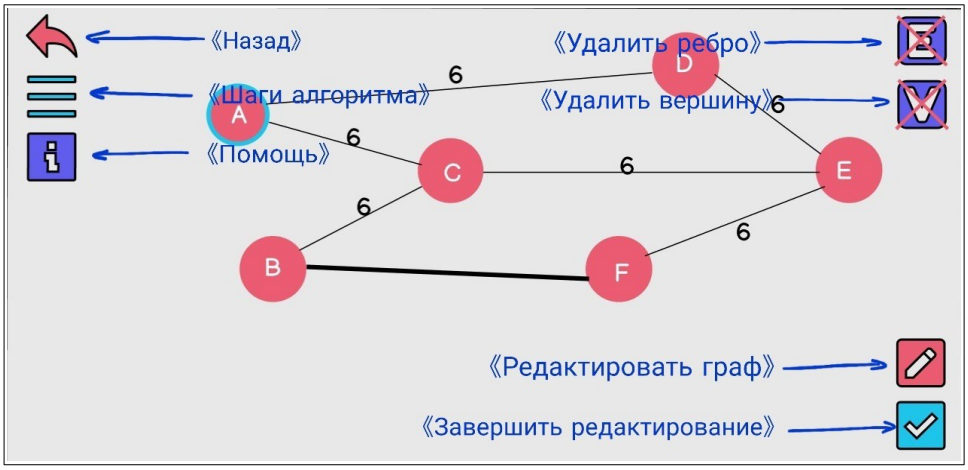

Рисунок 4 – Прототип интерфейса окна работы алгоритма

Рисунок 5 – Прототип интерфейса при нажатии кнопки «Помощь»

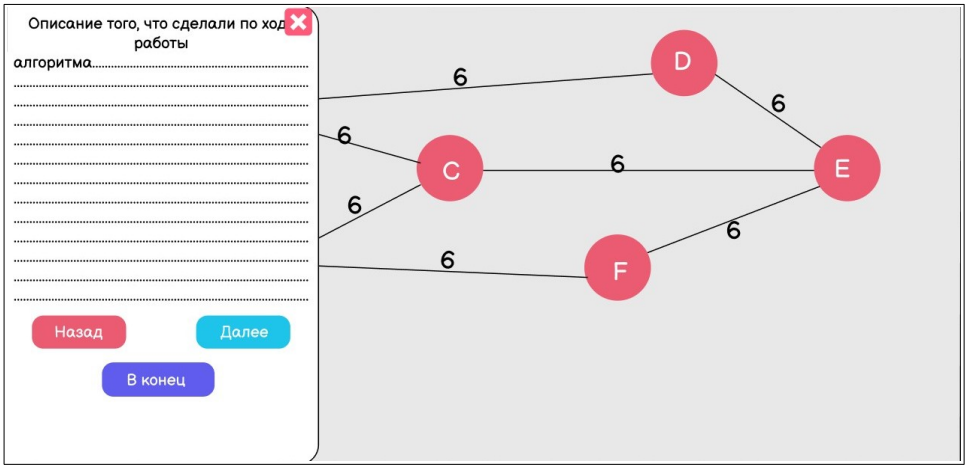

Рисунок 6 – Прототип интерфейса при нажатии кнопки «Шаги алгоритма»

Пользователь задаёт граф, добавляя новые вершины и рёбра в режиме редактирования.

Для того, чтобы добавить новую вершину, необходимо нажать на любое место на экране, а затем ввести имя вершины во всплывающем окне.

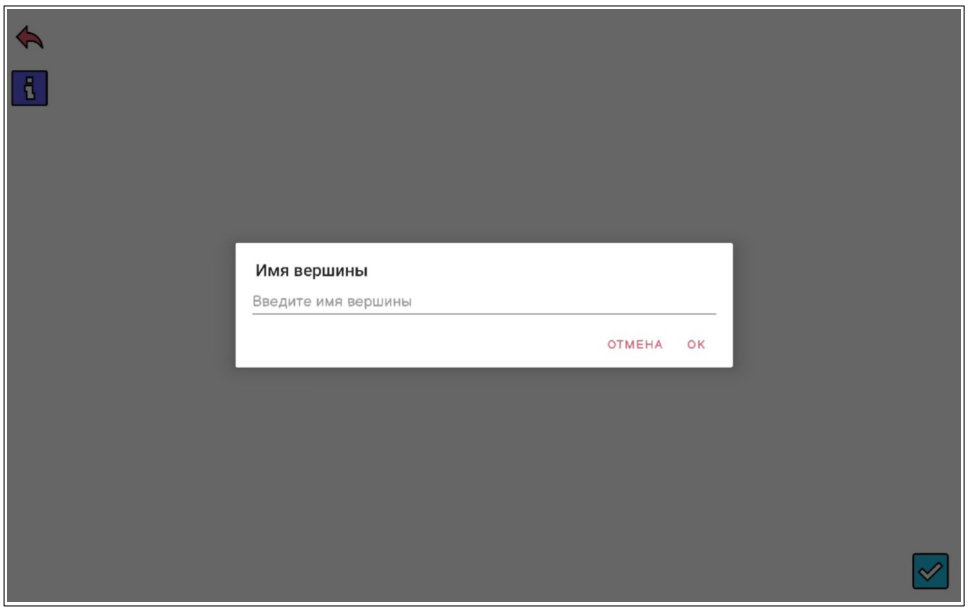

Рисунок 7 – Всплывающее окно для ввода имени новой вершины

При нажатии на уже существующую вершину, появляется кнопка «Удалить вершину».

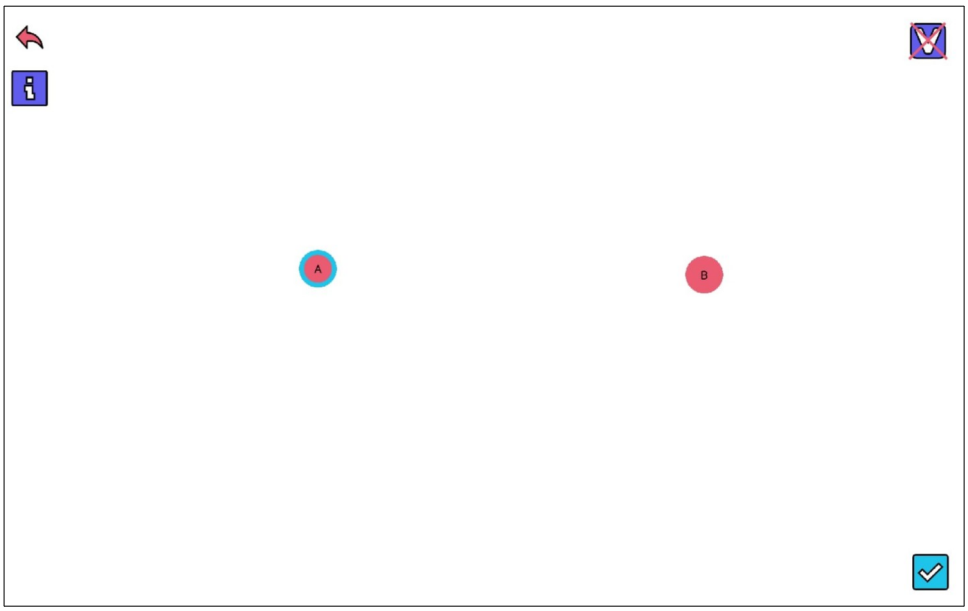

Рисунок 8 – Нажатие на уже существующую вершину

Чтобы добавить новое ребро, необходимо нажать сначала на начальную, затем на конечную вершину и ввести вес ребра во всплывающем окне.

Рисунок 9 – Всплывающее окно для ввода веса нового ребра

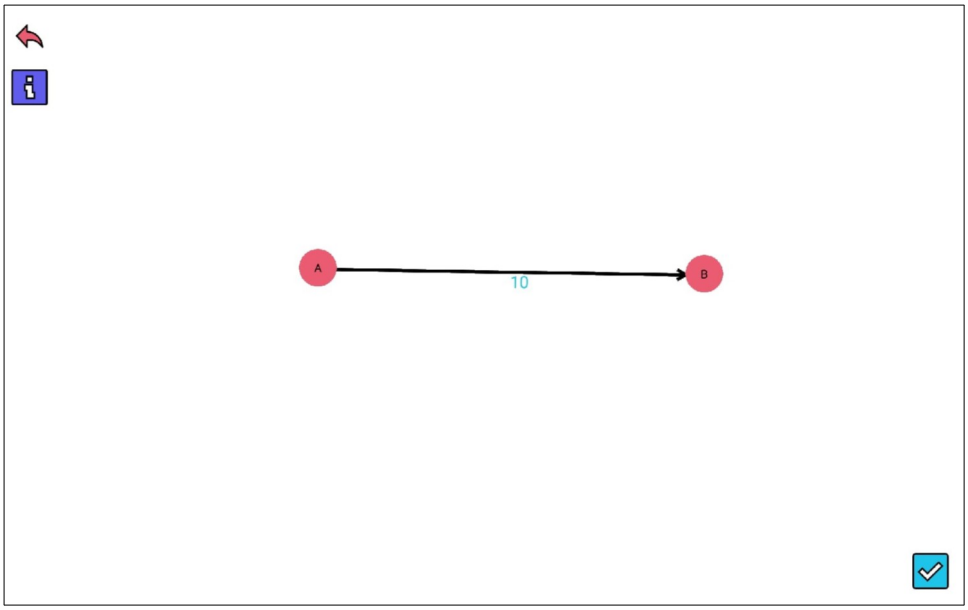

Рисунок 10 – Добавление нового ребра

При выборе уже существующего ребра, появляется кнопка «Удалить ребро».

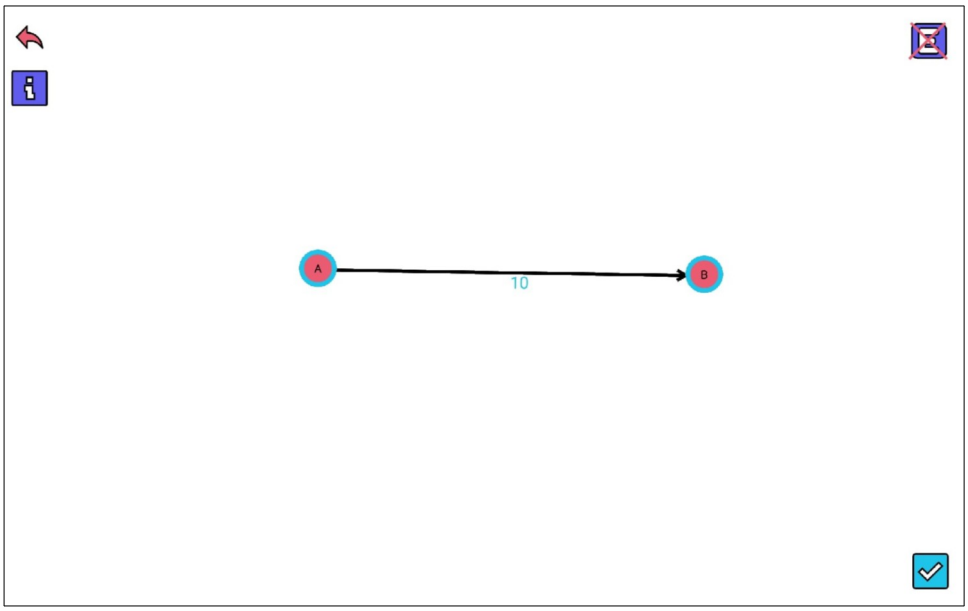

Рисунок 11 – Выбор уже существующего ребра

2. ПЛАН РАЗРАБОТКИ И РАСПРЕДЕЛЕНИЕ РОЛЕЙ

2.1. План разработки

До 02.07.2021 – Распределение по бригадам и выбор темы мини-проекта

До 05.07.2021 – Сдача вводного задания

До 07.07.2021 – Сдача прототипа графического интерфейса

До 09.07.2021 – Сдача первого этапа

До 12.07.2021 – Сдача второго этапа

До 14.07.2021 – Сдача финальной версии и отчёта

2.2. Распределение ролей в команде

- Камзолов Н.А. – лидер, фронтенд, ответственный за Android архитектуру.
- Гладких А.А. – алгоритмист, тестировщик.
- Моисейченко К.А. – алгоритмист, документация.

3. ОСОБЕННОСТИ РЕАЛИЗАЦИИ

3.1. Структуры данных

В программе использовались следующие структуры данных:

1. Структура Map — ассоциативный список в языке Kotlin – для хранения списка смежности.
2. Структура MutableList — изменяемый список — для хранения списка объектов, а также для хранения путей от стартовой вершины до всех оставшихся.

3.2. Основные методы

Файл Graph.kt – файл, в котором представлен класс, являющийся реализацией графа. Имеет поля vertexAmount и edgeAmount для хранения числа вершин и ребер соответственно, а также структуру данных Map для хранения списка смежности.

Файл BellmanFord.kt – файл, в котором реализован алгоритм Форда-Беллмана. Имеет метод runAlgorithm() для запуска алгоритма на заданном в конструкторе графе, а также другие методы, нужные для отображения хода работы алгоритма. Также, для удобства передачи информации о шагах алгоритма, был заведен класс Step, в который передается вся информация о шаге — его тип, который хранится в классе перечислении StepMsg, а также сопутствующая информация об изменениях, которая хранится в классе хранилище StepData.

Файл <Имя>Fragment.kt – файл, который представляет собой часть пользовательского интерфейса, здесь определяется, то как будет отображаться графический макет экрана *<Имя>*Screen, также файл отвечает за обновление UI и навигацию к другим фрагментам.

Файл <Имя>ViewModel.kt – файл, который представляет собой хранилище данных и логики текущего фрагмента. Он отвечает за события, связанные с нажатием кнопок и сообщает ***<Имя>***Fragment об необходимых обновлениях UI или о необходимой навигации.

Файл AlgorithmStepAdapter.kt – файл, который хранит в себе Адаптер для отображения информации о шаге алгоритма в RecyclerView(списке)**.**

Файл StructuresUtil.kt – файл, в котором хранятся data class-ы для хранения информации о графическом отображении графа.

Файл AppAnimation.kt – файл, в котором хранятся функции для исполнения некоторых анимаций, используемых в приложении.

Файл AppFullScreen.kt – файл, в котором хранится функция для полноэкранного отображения приложения.

Файл MainActivity.kt – основной файл приложения, создает само окно приложения и позволяет взаимодействовать с ним, осуществляя навигацию фрагментов.

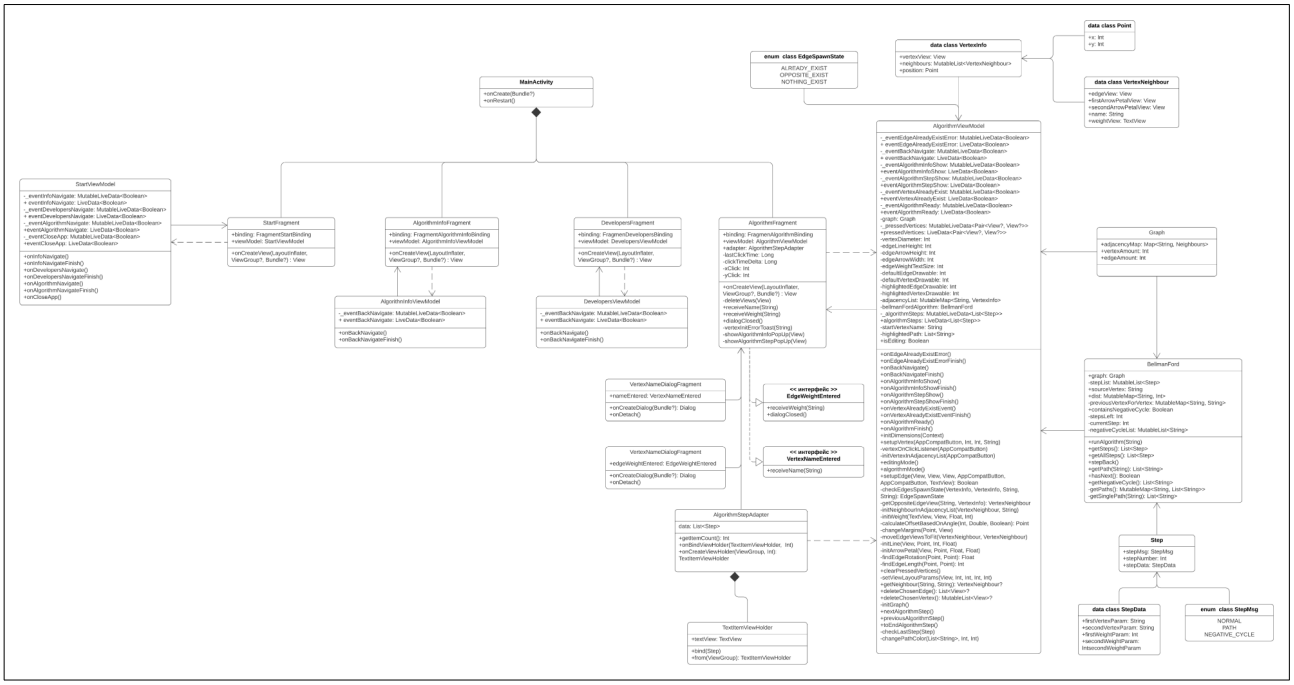

Рисунок 12 - UML диаграмма классов

Для удобства UML-диаграмма также была представлена отдельным файлом в папке с кодом.

4. ТЕСТИРОВАНИЕ

4.1. Тестирование класса Graph

Класс Graph был протестирован на корректность ввода данных. Здесь и далее для тестирования используется библиотека для модульного тестирования программного обеспечения Junit.

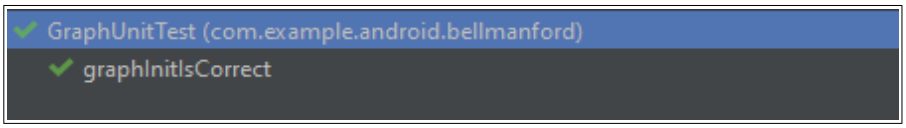

Рисунок 13 - иллюстрация корректного выполнения тестов класса Graph

4.2. Тестирование класса BellmanFord

Данный класс является реализацией алгоритма Форда-Беллмана. Сперва была протестирована корректная работа алгоритма на различных графах без отрицательного цикла, а затем отдельно была протестирована работа алгоритма на графе с отрицательным циклом. В обоих случаях программа успешно прошла все тесты.

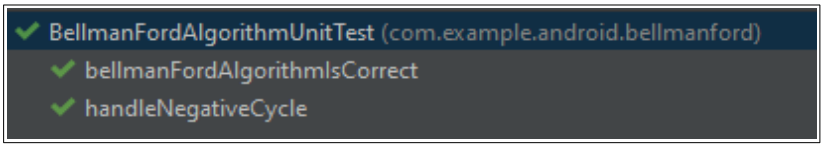

Рисунок 14 - иллюстрация корректного выполнения тестов класса BellmanFord

4.3. Тестирование UI

Было проведено тестирование UI. Все кнопки корректно отображаются, и нажатия на кнопки корректно срабатывают. Было проведено исследование удобства интерфейса — большая часть опрашиваемых с легкостью создали граф и запустили алгоритм. Единственный недостаток, который был отмечен малой частью контрольной группы — невозможность добавления ребра плавным движением от одной вершины к другой.

ЗАКЛЮЧЕНИЕ

Было реализовано приложение с графическим интерфейсом под операционную систему Android на языке Kotlin. Приложение пошагово демонстрирует работу алгоритма Форда-Беллмана. Был реализован интерфейс редактирования графа — в том числе возможность добавлять и удалять вершины и рёбра. Алгоритм Форда-Беллмана и класс, реализующий представление графа, были протестированы, и после тестирования программы алгоритм Форда-Беллмана работает корректно.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Алгоритм Форда-Беллмана. // MAXimal. URL: https://e-maxx.ru/algo/ford_bellman (дата обращения: 04.07.2021).

2. Kotlin Docs. // Kotlin. URL: https://kotlinlang.org/docs/home.html (дата обращения: 06.07.2021).

3. Введение в Kotlin JVM. // Stepik. URL: https://stepik.org/course/5448/syllabus (дата обращения: 01.07.2021).

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД

Экран AlgorithmScreen:

AlgorithmFragment.kt.

```
package com.example.android.bellmanford.algorithm

import android.annotation.SuppressLint
import android.os.Bundle
import android.view.*
import android.widget.*
import androidx.appcompat.widget.AppCompatButton
import androidx.core.content.ContextCompat
import androidx.databinding.DataBindingUtil
import androidx.fragment.app.Fragment
import androidx.lifecycle.ViewModelProvider
import androidx.navigation.fragment.findNavController
import com.example.android.bellmanford.R
import com.example.android.bellmanford.anim.AppAnimation
import com.example.android.bellmanford.databinding.AlgostepPopupBinding
import com.example.android.bellmanford.databinding.FragmentAlgorithmBinding
import com.example.android.bellmanford.dialogs.EdgeWeightDialogFragment
import com.example.android.bellmanford.dialogs.EdgeWeightEntered
import com.example.android.bellmanford.dialogs.VertexNameDialogFragment
import com.example.android.bellmanford.dialogs.VertexNameEntered
import com.example.android.bellmanford.util.AppFullscreen


class AlgorithmFragment : Fragment(), VertexNameEntered, EdgeWeightEntered {

    private lateinit var binding: FragmentAlgorithmBinding
    private lateinit var viewModel: AlgorithmViewModel
    private lateinit var adapter: AlgorithmStepAdapter

    private var lastClickTime = 0L
    private val clickTimeDelta = 500L

    private var xClick = 0
    private var yClick = 0

    @SuppressLint("ClickableViewAccessibility")
    override fun onCreateView(
        inflater: LayoutInflater, container: ViewGroup?,
        savedInstanceState: Bundle?
    ): View {

        binding = DataBindingUtil.inflate(
            layoutInflater,
            R.layout.fragment_algorithm, container, false
        )

        viewModel = ViewModelProvider(this).get(AlgorithmViewModel::class.java)

        adapter = AlgorithmStepAdapter()

        viewModel.algorithmSteps.observe(viewLifecycleOwner, {
            adapter.data = it
        })

        binding.algorithmViewModel = viewModel
```

```
        viewModel.eventBackNavigate.observe(viewLifecycleOwner, { event ->
            if (event) {
                findNavController().popBackStack()
                viewModel.onBackNavigateFinish()
                AppAnimation.fadingButtonAnimation(binding.btnBack)
            }
        })

        viewModel.eventAlgorithmStepShow.observe(viewLifecycleOwner, { event ->
            if (event) {
                showAlgorithmStepPopUp(binding.btnAlgoStep)
                viewModel.onAlgorithmStepShowFinish()
                AppAnimation.fadingButtonAnimation(binding.btnAlgoStep)
            }
        })

        viewModel.eventAlgorithmInfoShow.observe(viewLifecycleOwner, { event ->
            if (event) {
                showAlgorithmInfoPopUp(binding.btnAlgoInfo)
                viewModel.onAlgorithmInfoShowFinish()
                AppAnimation.fadingButtonAnimation(binding.btnAlgoInfo)
            }
        })

        binding.fragmentAlgorithmImgBtnEditingMode.setOnClickListener {
            viewModel.isEditing = true
            it.visibility = View.INVISIBLE
            binding.fragmentAlgorithmImgBtnAlgorithmMode.visibility = View.VISI-
BLE
            AppAnimation.fadingButtonAnimation(it)
            viewModel.editingMode()
        }

        binding.fragmentAlgorithmImgBtnAlgorithmMode.setOnClickListener {
            viewModel.isEditing = false
            it.visibility = View.INVISIBLE
            binding.fragmentAlgorithmImgBtnEditingMode.visibility = View.VISIBLE
            AppAnimation.fadingButtonAnimation(it)
            viewModel.algorithmMode()
                 Toast.makeText(requireContext(), "Выберите начальную вершину",
Toast.LENGTH_SHORT)
                .show()
        }

        viewModel.initDimensions(requireContext())

        binding.fragmentAlgorithmFltCanvas.setOnTouchListener { _, event ->
            if (event.action == MotionEvent.ACTION_DOWN) {
                xClick = event.x.toInt()
                yClick = event.y.toInt()
            }
            false
        }

        binding.fragmentAlgorithmFltCanvas.setOnClickListener {
            val curClickTime = System.currentTimeMillis()
            if(curClickTime - lastClickTime > clickTimeDelta) {
                if (viewModel.isEditing) {
                                                val  vertexNameDialogFragment  =
VertexNameDialogFragment(this)
                    activity?.let {
                        vertexNameDialogFragment.show(it.supportFragmentManager,
"New vertex")
                    }
                }
            }
```

```
            lastClickTime = curClickTime
        }

        viewModel.eventVertexAlreadyExist.observe(viewLifecycleOwner, {
            if (it) {
                    vertexInitErrorToast(getString(R.string.toast_explanation_al-
ready_exist))
                viewModel.onVertexAlreadyExistEventFinish()
            }
        })

        viewModel.eventAlgorithmReady.observe(viewLifecycleOwner, {
            if (it) binding.btnAlgoStep.visibility = View.VISIBLE
            else binding.btnAlgoStep.visibility = View.INVISIBLE
        })


        viewModel.pressedVertices.observe(viewLifecycleOwner, {
            if (it.first != null && it.second != null) {
                    binding.fragmentAlgorithmImgBtnDeleteVertex.visibility =
View.INVISIBLE
                val firstButton = it.first as AppCompatButton
                val secondButton = it.second as AppCompatButton
                val isEdgeAlreadyExist = viewModel.getNeighbour(
                    secondButton.text.toString(), firstButton.text.toString()
                )
                if (isEdgeAlreadyExist != null) {
                        binding.fragmentAlgorithmImgBtnDeleteEdge.visibility =
View.VISIBLE
                } else {
                                                val  edgeWeightDialogFragment  =
EdgeWeightDialogFragment(this)
                    activity?.let { temp ->
                          edgeWeightDialogFragment.show(temp.supportFragmentMan-
ager, "New edge")
                    }
                }
            } else if (it.first != null) {
                    binding.fragmentAlgorithmImgBtnDeleteVertex.visibility =
View.VISIBLE
                binding.fragmentAlgorithmImgBtnDeleteEdge.visibility = View.IN-
VISIBLE
            } else {
                    binding.fragmentAlgorithmImgBtnDeleteVertex.visibility =
View.INVISIBLE
                binding.fragmentAlgorithmImgBtnDeleteEdge.visibility = View.IN-
VISIBLE
            }
        })

        binding.fragmentAlgorithmImgBtnDeleteVertex.setOnClickListener {
            AppAnimation.fadingButtonAnimation(it)
            val viewsToDelete = viewModel.deleteChosenVertex()
            viewsToDelete?.let {
                it.forEach { view ->
                    deleteViews(view)
                }
            }
            viewModel.clearPressedVertices()
        }

        binding.fragmentAlgorithmImgBtnDeleteEdge.setOnClickListener {
            AppAnimation.fadingButtonAnimation(it)
            val viewsToDelete = viewModel.deleteChosenEdge()
            viewsToDelete?.let {
                it.forEach { view ->
```

```
                    deleteViews(view)
                }
            }
            viewModel.clearPressedVertices()
        }


        return binding.root
    }

    private fun deleteViews(view: View) {
        binding.fragmentAlgorithmFltCanvas.removeView(view)
    }

    override fun receiveName(name: String) {
        val newVertex = AppCompatButton(requireContext())
        if (name.isEmpty()) {
            vertexInitErrorToast(getString(R.string.toast_explanation_name_not_e
ntered))
            return
        }
        if (viewModel.setupVertex(newVertex, xClick, yClick, name)) {
            binding.fragmentAlgorithmFltCanvas.addView(newVertex)
        }


    }

    override fun receiveWeight(weight: String) {
        val newLine = View(requireContext())
        val firstArrowPetal = View(requireContext())
        val secondArrowPetal = View(requireContext())
        val edgeWeight = TextView(requireContext())
        if (weight.toIntOrNull() == null) {
            viewModel.clearPressedVertices()
            Toast.makeText(
                requireContext(),
                "Ребро не было создано, вес ребра не введен",
                Toast.LENGTH_SHORT
            ).show()
            return
        }
        else edgeWeight.text = weight.toInt().toString()
        viewModel.setupEdge(
            newLine,
            firstArrowPetal,
            secondArrowPetal,
            viewModel.pressedVertices.value?.second as AppCompatButton,
            viewModel.pressedVertices.value?.first as AppCompatButton,
            edgeWeight
        )

         edgeWeight.setTextColor(ContextCompat.getColor(requireContext(), R.col-
or.main_blue))


        binding.fragmentAlgorithmFltCanvas.addView(newLine)
        binding.fragmentAlgorithmFltCanvas.addView(firstArrowPetal)
        binding.fragmentAlgorithmFltCanvas.addView(secondArrowPetal)
        binding.fragmentAlgorithmFltCanvas.addView(edgeWeight)
        edgeWeight.bringToFront()

        viewModel.clearPressedVertices()
    }

    override fun dialogClosed() {
```

```
        viewModel.clearPressedVertices()
    }

    private fun vertexInitErrorToast(explanation: String) {
        Toast.makeText(
            context,
            getString(R.string.toast_text_vertex_was_not_spawned, explanation),
            Toast.LENGTH_SHORT
        ).show()
    }

    private fun showAlgorithmInfoPopUp(view: View) {

        val popupView: View = LayoutInflater.from(activity).inflate(R.layout.al-
goinfo_popup, null)

        val popupWindow = PopupWindow(
            popupView,
            LinearLayout.LayoutParams.WRAP_CONTENT,
            LinearLayout.LayoutParams.WRAP_CONTENT,
            true
        )

        val exitButton = popupView.findViewById<ImageButton>(R.id.exit_button)

        exitButton.setOnClickListener {
            if (popupWindow.isShowing) {
                AppAnimation.fadingButtonAnimation(it)
                popupWindow.dismiss()
            }
        }

        popupWindow.animationStyle = R.style.PopUpAnimationFromBottom
        popupWindow.showAtLocation(view, Gravity.CENTER, 0, 0)
        popupWindow.setOnDismissListener {
            AppFullscreen.turnFullscreen(requireActivity())
        }
    }

    private fun showAlgorithmStepPopUp(view: View) {
        val binding = DataBindingUtil.inflate<AlgostepPopupBinding>(
            layoutInflater,
            R.layout.algostep_popup, null, false
        )
        binding.popupAlgorithmStepRvStepExplanation.adapter = adapter

        val popupWindow = PopupWindow(
            binding.root,

            LinearLayout.LayoutParams.WRAP_CONTENT,
            LinearLayout.LayoutParams.MATCH_PARENT,
            true
        )


        binding.popupAlgorithmStepImgBtnClose.setOnClickListener {
            if (popupWindow.isShowing) {
                AppAnimation.fadingButtonAnimation(it)
                popupWindow.dismiss()
            }
        }

        binding.popupAlgorithmStepImgBtnPrevious.setOnClickListener {
            AppAnimation.fadingButtonAnimation(it)
            viewModel.previousAlgorithmStep()
        }
```

```
        binding.popupAlgorithmStepImgBtnNext.setOnClickListener {
            AppAnimation.fadingButtonAnimation(it)
            viewModel.nextAlgorithmStep()
        }

        binding.popupAlgorithmStepImgBtnToEnd.setOnClickListener {
            AppAnimation.fadingButtonAnimation(it)
            viewModel.toEndAlgorithmStep()
        }


        popupWindow.animationStyle = R.style.PopUpAnimationFromLeft
        popupWindow.showAtLocation(view, Gravity.START, 0, 0)

        popupWindow.setOnDismissListener {
            AppFullscreen.turnFullscreen(requireActivity())
        }
    }

}
```

AlgorithmViewModel.kt.

```
package com.example.android.bellmanford.algorithm

import android.content.Context
import android.util.TypedValue
import android.view.View
import android.widget.FrameLayout
import android.widget.TextView
import androidx.annotation.DrawableRes
import androidx.appcompat.widget.AppCompatButton
import androidx.core.view.marginLeft
import androidx.core.view.marginTop
import androidx.lifecycle.LiveData
import androidx.lifecycle.MutableLiveData
import androidx.lifecycle.ViewModel
import com.example.android.bellmanford.R
import java.lang.Math.*
import java.util.*
import kotlin.math.abs
import kotlin.math.atan
import kotlin.math.pow
import kotlin.math.sqrt

enum class EdgeSpawnStates {
    ALREADY_EXIST,
    OPPOSITE_EXIST,
    NOTHING_EXIST
}

class AlgorithmViewModel : ViewModel() {


    //region onClick events
    private val _eventEdgeAlreadyExistError = MutableLiveData<Boolean>()
    val eventEdgeAlreadyExistError: LiveData<Boolean>
        get() = _eventEdgeAlreadyExistError

    fun onEdgeAlreadyExistError() {
        _eventEdgeAlreadyExistError.value = true
    }

    fun onEdgeAlreadyExistErrorFinish() {
        _eventEdgeAlreadyExistError.value = false
    }
```

```
private val _eventBackNavigate = MutableLiveData<Boolean>()
val eventBackNavigate: LiveData<Boolean>
    get() = _eventBackNavigate

fun onBackNavigate() {
    _eventBackNavigate.value = true
}

fun onBackNavigateFinish() {
    _eventBackNavigate.value = false
}

private val _eventAlgorithmInfoShow = MutableLiveData<Boolean>()
val eventAlgorithmInfoShow: LiveData<Boolean>
    get() = _eventAlgorithmInfoShow

fun onAlgorithmInfoShow() {
    _eventAlgorithmInfoShow.value = true
}

fun onAlgorithmInfoShowFinish() {
    _eventAlgorithmInfoShow.value = false
}

private val _eventAlgorithmStepShow = MutableLiveData<Boolean>()
val eventAlgorithmStepShow: LiveData<Boolean>
    get() = _eventAlgorithmStepShow

fun onAlgorithmStepShow() {
    _eventAlgorithmStepShow.value = true
}

fun onAlgorithmStepShowFinish() {
    _eventAlgorithmStepShow.value = false
}

private val _eventVertexAlreadyExist = MutableLiveData<Boolean>()
val eventVertexAlreadyExist: LiveData<Boolean>
    get() = _eventVertexAlreadyExist


fun onVertexAlreadyExistEvent() {
    _eventVertexAlreadyExist.value = true
}

fun onVertexAlreadyExistEventFinish() {
    _eventVertexAlreadyExist.value = false
}

private val _eventAlgorithmReady = MutableLiveData<Boolean>()
val eventAlgorithmReady: LiveData<Boolean>
    get() = _eventAlgorithmReady

fun onAlgorithmReady() {
    _eventAlgorithmReady.value = true
}

fun onAlgorithmFinish() {
    _eventAlgorithmReady.value = false
}

//endregion

private lateinit var graph: Graph

private val _pressedVertices = MutableLiveData<Pair<View?, View?>>()
```

```
    val pressedVertices: LiveData<Pair<View?, View?>>
        get() = _pressedVertices


    private var vertexDiameter = 0
    private var edgeLineHeight = 0
    private var edgeArrowHeight = 0
    private var edgeArrowWidth = 0
    private var edgeWeightTextSize = 0
    @DrawableRes private var defaultEdgeDrawable = 0
    @DrawableRes private var defaultVertexDrawable = 0
    @DrawableRes private var highlightedEdgeDrawable = 0
    @DrawableRes private var highlightedVertexDrawable = 0

    private val adjacencyList = mutableMapOf<String, VertexInfo>()
    private lateinit var bellmanFordAlgorithm: BellmanFord

    private val _algorithmSteps = MutableLiveData<List<Step>>()
    val algorithmSteps: LiveData<List<Step>>
        get() = _algorithmSteps

    private var startVertexName = ""

    private var highlightedPath = listOf<String>()

    var isEditing = true

    fun initDimensions(context: Context) {
        vertexDiameter =
             context.resources.getDimension(R.dimen.size_fragment_algorithm_ver-
tex).toInt()
        edgeLineHeight =
            context.resources.getDimension(R.dimen.height_fragment_algorithm_edg
e).toInt()
        edgeArrowHeight =
            context.resources.getDimension(R.dimen.height_fragment_algorithm_ar-
row).toInt()
        edgeArrowWidth =
             context.resources.getDimension(R.dimen.width_fragment_algorithm_ar-
row).toInt()
        edgeWeightTextSize =
            context.resources.getDimension(R.dimen.text_size_fragment_algorithm_
edge_weight).toInt()
        defaultEdgeDrawable = R.drawable.view_line
        defaultVertexDrawable = R.drawable.img_graph_vertex
        highlightedEdgeDrawable = R.drawable.view_line_highlighted
        highlightedVertexDrawable = R.drawable.img_graph_vertex_selected
    }

    //region vertex creating
    fun setupVertex(
        vertexView: AppCompatButton,
        xClick: Int,
        yClick: Int,
        vertexName: String
    ): Boolean {
        setViewLayoutParams(
            vertexView,
            vertexDiameter,
            vertexDiameter,
            xClick - vertexDiameter / 2,
            yClick - vertexDiameter / 2
        )
        vertexView.isClickable = true
        vertexView.setBackgroundResource(R.drawable.img_graph_vertex)
```

```
                                vertexView.text    =    vertexName.replace("    ",
"").toUpperCase(Locale.ROOT).take(4)
        if (adjacencyList.containsKey(vertexView.text)) {
            onVertexAlreadyExistEvent()
            return false
        }
        initVertexInAdjacencyList(vertexView)

        vertexView.setOnClickListener {
            vertexOnClickListener(it as AppCompatButton)
        }
        return true
    }

    private fun vertexOnClickListener(vertexView: AppCompatButton) {
        if(isEditing) {
            val tempPair = _pressedVertices.value ?: Pair(null, null)
            when (vertexView) {
                tempPair.first -> {
                      vertexView.setBackgroundResource(R.drawable.img_graph_ver-
tex)
                    _pressedVertices.value = Pair(tempPair.second, null)
                }
                tempPair.second -> {
                      vertexView.setBackgroundResource(R.drawable.img_graph_ver-
tex)
                    _pressedVertices.value = Pair(tempPair.first, null)
                }

                else -> {
                      vertexView.setBackgroundResource(R.drawable.img_graph_ver-
tex_selected)
                    tempPair.second?.setBackgroundResource(R.drawable.img_graph_
vertex)
                    _pressedVertices.value = Pair(vertexView, tempPair.first)

                }
            }
        }
        else {
            if(startVertexName == "") {
                initGraph()
                bellmanFordAlgorithm = BellmanFord(graph)
                bellmanFordAlgorithm.runAlgorithm(vertexView.text.toString())
                startVertexName = vertexView.text.toString()
                vertexView.setBackgroundResource(highlightedVertexDrawable)
                onAlgorithmReady()
                nextAlgorithmStep()
            }
        }
    }

    private fun initVertexInAdjacencyList(vertex: AppCompatButton) {
        val params = vertex.layoutParams as FrameLayout.LayoutParams
        val center =
               Point(params.leftMargin + vertexDiameter / 2, params.topMargin +
vertexDiameter / 2)
             adjacencyList[vertex.text.toString()] = VertexInfo(vertex, muta-
bleListOf(), center)
    }
    //endregion

    fun editingMode() {
           adjacencyList[startVertexName]?.vertexView?.setBackgroundResource(de-
faultVertexDrawable)
        startVertexName = ""
```

```
        changePathColor(highlightedPath, defaultVertexDrawable, defaultEdgeDrawable)
        _algorithmSteps.value = listOf()
        onAlgorithmFinish()
    }

    fun algorithmMode() {
        clearPressedVertices()
    }

    //region edge creating
    fun setupEdge(
        edgeView: View,
        firstArrowPetalView: View,
        secondArrowPetalView: View,
        firstVertexView: AppCompatButton,
        secondVertexView: AppCompatButton,
        edgeWeightView: TextView
    ): Boolean {
         val vertexInfo1: VertexInfo = adjacencyList.getOrElse(firstVertexView.text.toString(), {
            return false
        })
        val vertexInfo2: VertexInfo = adjacencyList.getOrElse(secondVertexView.text.toString(), {
            return false
        })

        val point1 = vertexInfo1.position
        val point2 = vertexInfo2.position

        val length = findEdgeLength(point1, point2)
        val rotation = findEdgeRotation(point1, point2)

        val edgeSpawnState =
            checkEdgesSpawnState(
                vertexInfo1,
                vertexInfo2,
                firstVertexView.text.toString(),
                secondVertexView.text.toString()
            )

        if (edgeSpawnState == EdgeSpawnStates.ALREADY_EXIST) return false

        initLine(edgeView, point1, length, rotation)
        initArrowPetal(firstArrowPetalView, point2, rotation, 30F)
        initArrowPetal(secondArrowPetalView, point2, rotation, -30F)
        initWeight(edgeWeightView, edgeView, rotation, length)

        val newVertexNeighbour = VertexNeighbour(
            edgeView,
            firstArrowPetalView,
            secondArrowPetalView,
            secondVertexView.text.toString(),
            edgeWeightView
        )

         initNeighbourInAdjacencyList(newVertexNeighbour, firstVertexView.text.toString())
        println(adjacencyList)

        if (edgeSpawnState == EdgeSpawnStates.OPPOSITE_EXIST) {
                            getOppositeEdgeView(firstVertexView.text.toString(), vertexInfo2)?.let {
                moveEdgeViewsToFit(newVertexNeighbour, it)
            }
```

```
        }
        return true
    }

    private fun checkEdgesSpawnState(
        vertexInfo1: VertexInfo,
        vertexInfo2: VertexInfo,
        firstVertexName: String,
        secondVertexName: String
    ): EdgeSpawnStates {
        vertexInfo1.neighbours.forEach {
            if (it.name == secondVertexName)
                return EdgeSpawnStates.ALREADY_EXIST
        }
        vertexInfo2.neighbours.forEach {
            if (it.name == firstVertexName)
                return EdgeSpawnStates.OPPOSITE_EXIST
        }
        return EdgeSpawnStates.NOTHING_EXIST
    }

    private fun getOppositeEdgeView(
        oppositeVertexName: String,
        vertexInfo: VertexInfo
    ): VertexNeighbour? {
        vertexInfo.neighbours.forEach {
            if (it.name == oppositeVertexName) {
                return it
            }
        }
        return null
    }

    private fun initNeighbourInAdjacencyList(
        vertexNeighbour: VertexNeighbour,
        firstVertexName: String
    ) {
        adjacencyList[firstVertexName]?.neighbours?.add(vertexNeighbour)
    }

    private fun initWeight(
        edgeWeightView: TextView,
        edgeView: View,
        rotation: Float,
        edgeLength: Int
    ) {
        val offset = calculateOffsetBasedOnAngle(edgeLength / 2, rotation.toDou-
ble(), false)

        setViewLayoutParams(
            edgeWeightView,
            FrameLayout.LayoutParams.WRAP_CONTENT,
            FrameLayout.LayoutParams.WRAP_CONTENT,
            edgeView.marginLeft + offset.x,
            edgeView.marginTop + offset.y
        )

        edgeWeightView.pivotX = 0F
        edgeWeightView.isClickable = true

        if (abs(rotation) > 90) edgeWeightView.rotation = 180 + rotation
        else edgeWeightView.rotation = rotation

          edgeWeightView.setTextSize(TypedValue.COMPLEX_UNIT_PX, edgeWeightText-
Size.toFloat())
    }
```

```
    private fun calculateOffsetBasedOnAngle(
        radius: Int,
        rotation: Double,
        inverse: Boolean
    ): Point {
        val offsetX = radius * kotlin.math.cos(toRadians(rotation))
        val offsetY = radius * kotlin.math.sin(toRadians(rotation))
        return when (inverse) {
            false -> Point(offsetX.toInt(), offsetY.toInt())
            true -> Point(offsetY.toInt(), offsetX.toInt())
        }
    }

    private fun changeMargins(point: Point, view: View) {
        val params = view.layoutParams as FrameLayout.LayoutParams
        params.setMargins(
            point.x,
            point.y,
            0,
            0
        )
    }


     private fun moveEdgeViewsToFit(firstEdge: VertexNeighbour, secondEdge: Ver-
texNeighbour) {
        val firstOffset = calculateOffsetBasedOnAngle(
            vertexDiameter * 3 / 8,
            firstEdge.edgeView.rotation.toDouble(),
            true
        )
        val secondOffset = calculateOffsetBasedOnAngle(
            vertexDiameter * 3 / 8,
            secondEdge.edgeView.rotation.toDouble(),
            true
        )

        //move lines
        changeMargins(
            Point(
                firstEdge.edgeView.marginLeft - firstOffset.x,
                firstEdge.edgeView.marginTop + firstOffset.y
            ), firstEdge.edgeView
        )

        changeMargins(
            Point(
                secondEdge.edgeView.marginLeft - secondOffset.x,
                secondEdge.edgeView.marginTop + secondOffset.y
            ), secondEdge.edgeView
        )


        //move arrow
        changeMargins(
            Point(
                firstEdge.firstArrowPetalView.marginLeft - firstOffset.x,
                firstEdge.firstArrowPetalView.marginTop + firstOffset.y,
            ), firstEdge.firstArrowPetalView
        )

        changeMargins(
            Point(
                secondEdge.firstArrowPetalView.marginLeft - secondOffset.x,
                secondEdge.firstArrowPetalView.marginTop + secondOffset.y,
```

```
            ), secondEdge.firstArrowPetalView
        )

        changeMargins(
            Point(
                firstEdge.secondArrowPetalView.marginLeft - firstOffset.x,
                firstEdge.secondArrowPetalView.marginTop + firstOffset.y,
            ), firstEdge.secondArrowPetalView
        )

        changeMargins(
            Point(
                secondEdge.secondArrowPetalView.marginLeft - secondOffset.x,
                secondEdge.secondArrowPetalView.marginTop + secondOffset.y,
            ), secondEdge.secondArrowPetalView
        )

        //move weight
        changeMargins(
            Point(
                firstEdge.weightView.marginLeft - firstOffset.x,
                firstEdge.weightView.marginTop + firstOffset.y,
            ), firstEdge.weightView
        )

        changeMargins(
            Point(
                secondEdge.weightView.marginLeft - secondOffset.x,
                secondEdge.weightView.marginTop + secondOffset.y,
            ), secondEdge.weightView
        )
    }


    private fun initLine(edge: View, point: Point, length: Int, rotation: Float)
{
        setViewLayoutParams(
            edge,
            length,
               edge.context.resources.getDimension(R.dimen.height_fragment_algo-
rithm_edge).toInt(),
            point.x,
            point.y
        )
        edge.isClickable = true
        edge.pivotX = 0F
         edge.pivotY = edge.resources.getDimension(R.dimen.height_fragment_algo-
rithm_edge) / 2
        edge.rotation = rotation


        edge.setBackgroundResource(R.drawable.view_line)
    }

    private fun initArrowPetal(
        arrowPetal: View,
        point: Point,
        rotation: Float,
        additionalRotation: Float
    ) {
        val offset =
            calculateOffsetBasedOnAngle(vertexDiameter / 2, rotation.toDouble(),
false)
        setViewLayoutParams(
            arrowPetal,
            edgeArrowWidth,
```

```
            edgeArrowHeight,
            point.x - offset.x,
            point.y - offset.y
        )
        arrowPetal.pivotY = (edgeArrowHeight / 2).toFloat()
        arrowPetal.pivotX = 0F
        arrowPetal.rotation = rotation + additionalRotation + 180
        arrowPetal.setBackgroundResource(R.drawable.view_line)
    }


    private fun findEdgeRotation(point1: Point, point2: Point): Float {
        val triangleEdgeLength1 = abs(point1.x - point2.x).toDouble()
        val triangleEdgeLength2 = abs(point1.y - point2.y).toDouble()

        val angleTan = triangleEdgeLength2 / triangleEdgeLength1
        val angle = toDegrees(atan(angleTan))
        return when {
            //Quadrant 1
            (point2.x > point1.x && point2.y < point1.y) -> -angle.toFloat()
            //Quadrant 2
                 (point2.x < point1.x && point2.y < point1.y) -> -180 + an-
gle.toFloat()
            //Quadrant 3
                  (point2.x < point1.x && point2.y > point1.y) -> 180 - an-
gle.toFloat()
            //Quadrant 4
            else -> angle.toFloat()

        }
    }

    private fun findEdgeLength(point1: Point, point2: Point): Int {
        val triangleEdgeLength1 = abs(point1.x - point2.x).toDouble()
        val triangleEdgeLength2 = abs(point1.y - point2.y).toDouble()
                                  return    sqrt(triangleEdgeLength1.pow(2)    +
triangleEdgeLength2.pow(2)).toInt()
    }
    //endregion


    fun clearPressedVertices() {
        val tempPair = _pressedVertices.value ?: Pair(null, null)
        _pressedVertices.value = Pair(null, null)
        tempPair.first?.setBackgroundResource(R.drawable.img_graph_vertex)
        tempPair.second?.setBackgroundResource(R.drawable.img_graph_vertex)
    }

    private fun setViewLayoutParams(view: View, width: Int, height: Int, x: Int,
y: Int) {
        val params = FrameLayout.LayoutParams(
            width,
            height
        )
        params.setMargins(x, y, 0, 0)
        view.layoutParams = params
    }

    fun getNeighbour(
        firstVertexName: String,
        secondVertexName: String
    ): VertexNeighbour? {
        adjacencyList[firstVertexName]?.let {
            it.neighbours.forEach { neighbour ->
                if (neighbour.name == secondVertexName) {
                    return neighbour
```

```
                }
            }
        }
        return null
    }


    //region delete graph components
    fun deleteChosenEdge(): List<View>? {
        pressedVertices.value?.let {
            if (it.second != null && it.first != null) {
                val firstButton = it.first as AppCompatButton
                val secondButton = it.second as AppCompatButton
                val neighbour = getNeighbour(
                    secondButton.text.toString(),
                    firstButton.text.toString()
                )
                if (neighbour != null) {
                    adjacencyList[secondButton.text.toString()]?.neighbours?.re-
move(neighbour)
                    println(adjacencyList)
                    return listOf(
                        neighbour.edgeView,
                        neighbour.firstArrowPetalView,
                        neighbour.secondArrowPetalView,
                        neighbour.weightView
                    )
                }
            }
        }
        return null
    }

    fun deleteChosenVertex(): MutableList<View>? {

        pressedVertices.value?.let {
            if (it.first != null) {
                val vertexView = it.first as AppCompatButton
                val viewsList = mutableListOf<View>()
                adjacencyList[vertexView.text.toString()]?.neighbours?.forEach {
neighbour ->
                    viewsList.add(neighbour.edgeView)
                    viewsList.add(neighbour.firstArrowPetalView)
                    viewsList.add(neighbour.secondArrowPetalView)
                    viewsList.add(neighbour.weightView)
                }
                viewsList.add(vertexView)
                adjacencyList.remove(vertexView.text.toString())
                println(adjacencyList)
                return viewsList
            }

        }
        return null
    }
    //endregion

    private fun initGraph() {
        val algorithmAdjacencyList = mutableMapOf<String, Neighbours>()
        adjacencyList.forEach {
            algorithmAdjacencyList[it.key] = mutableListOf()

            it.value.neighbours.forEach { vertexNeighbour ->
                algorithmAdjacencyList[it.key]?.add(
                    Pair(
                        vertexNeighbour.name,
```

```
                    vertexNeighbour.weightView.text.toString().toInt()
                )
            )
        }

    }
    graph = Graph(algorithmAdjacencyList)
}

fun nextAlgorithmStep() {
    if(!bellmanFordAlgorithm.hasNext()) {
        changePathColor(
            highlightedPath,
            defaultVertexDrawable,
            defaultEdgeDrawable
        )
        return
    }
    val steps = bellmanFordAlgorithm.getSteps()
    _algorithmSteps.value = steps
    checkLastStep(steps.last())
}

fun previousAlgorithmStep() {
    bellmanFordAlgorithm.stepBack()
    bellmanFordAlgorithm.stepBack()
    nextAlgorithmStep()
}

fun toEndAlgorithmStep() {
    val steps = bellmanFordAlgorithm.getAllSteps()
    _algorithmSteps.value = steps
    checkLastStep(steps.last())
}

private fun checkLastStep(lastStep: Step) {
    changePathColor(
        highlightedPath,
        defaultVertexDrawable,
        defaultEdgeDrawable
    )
    if(lastStep.stepMsg == StepMsg.PATH) {
        highlightedPath = bellmanFordAlgorithm.getPath(
            lastStep.stepData.secondVertexParam
        )
        changePathColor(
            highlightedPath,
            highlightedVertexDrawable,
            highlightedEdgeDrawable
        )
    }

    if(lastStep.stepMsg == StepMsg.NEGATIVE_CYCLE) {
        changePathColor(
            highlightedPath,
            defaultVertexDrawable,
            defaultEdgeDrawable
        )
        highlightedPath = bellmanFordAlgorithm.getNegativeCycle()
        changePathColor(
            highlightedPath,
            highlightedVertexDrawable,
            highlightedEdgeDrawable
        )
    }
}
```

```
    private fun changePathColor(path: List<String>, @DrawableRes vertexDrawable:
Int, @DrawableRes edgeDrawable: Int) {
        for(i in 0..path.size - 2) {
               adjacencyList[path[i]]?.vertexView?.setBackgroundResource(vertex-
Drawable)
            val edge = getNeighbour(path[i], path[i+1])
            edge?.firstArrowPetalView?.setBackgroundResource(edgeDrawable)
            edge?.secondArrowPetalView?.setBackgroundResource(edgeDrawable)
            edge?.edgeView?.setBackgroundResource(edgeDrawable)
        }
        if(path.isNotEmpty())
              adjacencyList[path.last()]?.vertexView?.setBackgroundResource(ver-
texDrawable)
    }
}
```

AlgorithmStepAdapter.kt.

```
package com.example.android.bellmanford.algorithm

import android.view.LayoutInflater
import android.view.ViewGroup
import android.widget.TextView
import androidx.recyclerview.widget.RecyclerView
import com.example.android.bellmanford.R

class AlgorithmStepAdapter : RecyclerView.Adapter<AlgorithmStepAdapter.Tex-
tItemViewHolder>() {

    var data = listOf<Step>()
        set(value) {
            field = value
            notifyDataSetChanged()
        }

    override fun getItemCount() = data.size

    override fun onBindViewHolder(holder: TextItemViewHolder, position: Int) {
        val item = data[position]
        holder.bind(item)
    }

      override fun onCreateViewHolder(parent: ViewGroup, viewType: Int): Tex-
tItemViewHolder {
        return TextItemViewHolder.from(parent)
    }

                class TextItemViewHolder(val textView: TextView) :
RecyclerView.ViewHolder(textView) {

        fun bind(item: Step) {
            when (item.stepMsg) {
                StepMsg.NORMAL -> {
                    val firstWeightParams = if(item.stepData.firstWeightParam !=
Int.MAX_VALUE) item.stepData.firstWeightParam.toString() else "∞"
                    textView.text = textView.context.resources.getString(
                        R.string.txt_algorithm_step_default,
                        item.stepNumber,
                        item.stepData.firstVertexParam,
                        item.stepData.secondVertexParam,
                        item.stepData.secondVertexParam,
                        firstWeightParams,
                        item.stepData.firstVertexParam,
                        item.stepData.secondVertexParam,
                        item.stepData.secondWeightParam
                    )
                }
```

```
                StepMsg.NEGATIVE_CYCLE -> {
                    textView.text =
                        textView.context.resources.getString(
                            R.string.txt_algorithm_step_negative_cycle,
                            item.stepNumber)
                }

                StepMsg.PATH -> {
                    if(item.stepData.firstWeightParam != Int.MAX_VALUE) {
                        textView.text = textView.context.resources.getString(
                            R.string.txt_algorithm_path,
                            item.stepNumber,
                            item.stepData.firstVertexParam,
                            item.stepData.secondVertexParam,
                            item.stepData.firstWeightParam
                        )
                    }
                    else {
                        textView.text = textView.context.resources.getString(
                            R.string.txt_algorithm_no_path,
                            item.stepNumber,
                            item.stepData.firstVertexParam,
                            item.stepData.secondVertexParam
                        )
                    }
                }
            }
        }

        companion object {
            fun from(parent: ViewGroup): TextItemViewHolder {
                val layoutInflater = LayoutInflater.from(parent.context)
                val view = layoutInflater
                    .inflate(R.layout.text_item_view, parent, false) as TextView

                return TextItemViewHolder(view)
            }
        }
    }
}
```

BellmanFord.kt.

```
package com.example.android.bellmanford.algorithm


data class StepData(
    val firstVertexParam: String,
    val secondVertexParam: String,
    val firstWeightParam: Int,
    val secondWeightParam: Int?
)

enum class StepMsg{
    NORMAL,
    PATH,
    NEGATIVE_CYCLE
}

class Step {
    constructor(stepNumber: Int, stepMsg: StepMsg) {
        this.stepNumber = stepNumber
        this.stepMsg = stepMsg
    }
    constructor(stepNumber: Int, stepMsg: StepMsg, stepData: StepData) {
        this.stepNumber = stepNumber
        this.stepMsg = stepMsg
```

```
        this.stepData = stepData
    }

    var stepNumber = -1
    lateinit var stepMsg: StepMsg
    lateinit var stepData: StepData
}

// A class to represent a connected, directed and weighted graph
class BellmanFord(private val graph: Graph) {

    private val stepList = mutableListOf<Step>()
    var sourceVertex = ""
    val dist = mutableMapOf<String, Int>()
    private  val previousVertexForVertex = mutableMapOf<String, String>()
    var containsNegativeCycle = false

    private var stepsLeft = graph.adjacencyMap.size
    private var currentStep = 0
    private val negativeCycleList = mutableListOf<String>()

    fun runAlgorithm(src: String) {
        dist.clear()
        negativeCycleList.clear()
        currentStep = 0
        stepsLeft = 0
        sourceVertex = src

        // Step 1: Initialize distances from src to all other vertexes
        graph.adjacencyMap.forEach {
            dist[it.key] = Int.MAX_VALUE
        }
        dist[src] = 0
        previousVertexForVertex[src] = src

        // Step 2: Relax all edges |V| - 1 times
        repeat(graph.vertexAmount - 1) {
            graph.adjacencyMap.forEach {
                it.value.forEach { neighbour ->

                            val distanceToFirstNode = if (dist[it.key] != null)
dist[it.key]!! else 0
                        val distanceForSecondNode = if (dist[neighbour.first] !=
null) dist[neighbour.first]!! else 0

                    if(distanceToFirstNode != Int.MAX_VALUE &&
                        distanceToFirstNode + neighbour.second < distanceForSec-
ondNode) {
                            val newStep = Step(
                                currentStep + 1,
                                StepMsg.NORMAL,
                                StepData(it.key, neighbour.first,
                                    distanceForSecondNode,
                                    distanceToFirstNode + neighbour.second))
                            stepList.add(newStep)
                            currentStep++
                                dist[neighbour.first] = dist[it.key]!! + neigh-
bour.second
                            previousVertexForVertex[neighbour.first] = it.key
                    }
                }
            }
        }

        //Step 3: Check for negative cycle
        graph.adjacencyMap.forEach {
```

```
            it.value.forEach { neighbour ->

                            val distanceToFirstNode = if (dist[it.key] != null)
dist[it.key]!! else 0
                    val distanceForSecondNode = if (dist[neighbour.first] != null)
dist[neighbour.first]!! else 0

                if(distanceToFirstNode != Int.MAX_VALUE &&
                        distanceToFirstNode + neighbour.second < distanceForSec-
ondNode) {
                    val newStep = Step(
                        currentStep + 1,
                        StepMsg.NEGATIVE_CYCLE
                    )

                    var currentVertex : String? = it.key

                    repeat(graph.vertexAmount - 1) {
                        currentVertex = previousVertexForVertex[currentVertex]
                    }

                    var cycleVertex = currentVertex

                    while(cycleVertex != previousVertexForVertex[currentVertex])
{
                        previousVertexForVertex[currentVertex]?.let{ it -> nega-
tiveCycleList.add(it)}
                        currentVertex = previousVertexForVertex[currentVertex]!!
                    }
                    cycleVertex?.let { it1 -> negativeCycleList.add(it1) }


                    stepList.add(newStep)
                    currentStep = 0
                    stepsLeft = stepList.size
                    containsNegativeCycle = true
                    return
                }
            }
        }

        graph.adjacencyMap.forEach {
            val newStep = Step(
                currentStep + 1,
                StepMsg.PATH,
                StepData(sourceVertex,
                it.key,
                dist[it.key] ?: 0,
                null)
            )
            currentStep++
            stepList.add(newStep)
        }

        currentStep = 0
        stepsLeft = stepList.size
    }

    fun getSteps(): List<Step> {
           println("Trying to step : curStep ${currentStep}, stepsLeft ${step-
sLeft}")
        println("After doing step: curStep ${currentStep + 1}, stepsLeft ${step-
sLeft - 1}")
        stepsLeft--
        return stepList.slice(IntRange(0, currentStep++))
    }
```

```
    fun getAllSteps(): List<Step> {
        currentStep = stepList.size
        stepsLeft = 0
        return stepList
    }

    fun stepBack() {
         println("Trying to step back: curStep ${currentStep}, stepsLeft ${step-
sLeft}")
        if(currentStep > 0) {
            stepsLeft++
            currentStep--
        }
           println("After step back: curStep ${currentStep}, stepsLeft ${step-
sLeft}")
    }

    fun getPath(vertexName: String): List<String> {
        return getSinglePath(vertexName)
    }

    fun hasNext(): Boolean {
        println("In hasNext: stepsLeft=${stepsLeft}")
        return stepsLeft > 0
    }

    fun getNegativeCycle(): List<String> {
        negativeCycleList.add(negativeCycleList.first())
        return negativeCycleList.reversed()
    }

    private fun getPaths(): MutableMap<String, List<String>> {
        val paths = mutableMapOf<String, List<String>>()
        graph.adjacencyMap.forEach {
            paths[it.key] = getSinglePath(it.key)
        }

        return paths
    }

    private fun getSinglePath(vertexTo: String): List<String> {
        val path = mutableListOf<String>()
        path.add(vertexTo)

        var currentVertex = previousVertexForVertex[vertexTo]
        while(currentVertex != sourceVertex && currentVertex != vertexTo && cur-
rentVertex != null && !containsNegativeCycle) {
            path.add(currentVertex.toString())
            currentVertex = previousVertexForVertex[currentVertex]
        }

        if(currentVertex != null)
            path.add(currentVertex.toString())

        return path.reversed()
    }
}
```

Graph.kt

```
package com.example.android.bellmanford.algorithm

typealias Neighbour = Pair<String, Int>
typealias Neighbours = MutableList<Neighbour>


class Graph(val adjacencyMap: Map<String, Neighbours>) {
```

```
    // An inner class to represent a weighted edge in graph
    inner class Edge(var src: String, var dest: String, var weight: Double)

    var vertexAmount = adjacencyMap.size
    var edgeAmount = 0
    init {
        adjacencyMap.forEach {
            edgeAmount += it.value.size
        }
    }
}
```

StructuresUtil.kt

```
package com.example.android.bellmanford.algorithm

import android.view.View
import android.widget.TextView


data class VertexNeighbour(
    val edgeView: View,
    val firstArrowPetalView: View,
    val secondArrowPetalView: View,
    val name: String,
    val weightView: TextView
)

data class Point(
    val x: Int,
    val y: Int
)

data class VertexInfo(
    val vertexView: View,
    val neighbours: MutableList<VertexNeighbour>,
    val position: Point
)
```

algoinfo_popup.kt

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout        xmlns:android="http://
schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content">

    <LinearLayout
        android:id="@+id/linearLayout2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:background="@color/background_grey"
        android:gravity="center"
        android:orientation="vertical"
        android:paddingLeft="10dp"
        android:paddingRight="10dp"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent">

        <LinearLayout
            android:layout_width="wrap_content"
            android:layout_height="250dp"
            android:orientation="vertical"
            android:padding="10dp">
```

```
            <pl.droidsonroids.gif.GifImageView
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:src="@drawable/infogif" />

        </LinearLayout>
    </LinearLayout>

    <ImageButton
        android:id="@+id/exit_button"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:background="@drawable/btn_popup_exit"

        app:layout_constraintEnd_toEndOf="@id/linearLayout2"
        app:layout_constraintTop_toTopOf="parent">


    </ImageButton>
</androidx.constraintlayout.widget.ConstraintLayout>
```

algoinfo_popup.kt

```
<?xml version="1.0" encoding="utf-8"?>
<layout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto">

    <LinearLayout
        android:layout_width="wrap_content"
        android:layout_height="match_parent"
        android:background="@color/background_grey"
        android:orientation="vertical">

        <ImageButton
            android:id="@+id/popup_algorithm_step_img_btn_close"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_gravity="end"
            android:layout_marginTop="@dimen/margin_15dp"
            android:layout_marginEnd="@dimen/margin_15dp"
            android:layout_marginBottom="@dimen/margin_15dp"
            android:background="@drawable/btn_popup_exit" />


        <androidx.recyclerview.widget.RecyclerView
            android:id="@+id/popup_algorithm_step_rv_step_explanation"
            android:layout_width="300dp"
            android:layout_height="0dp"
            android:layout_marginStart="5dp"
            android:layout_marginEnd="5dp"
            android:layout_weight="1"
            android:background="@drawable/view_rv_background"
            app:layoutManager="androidx.recyclerview.widget.LinearLayoutManager"
/>

        <LinearLayout
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_gravity="center"
            android:layout_marginTop="@dimen/margin_15dp"
            android:orientation="horizontal">

            <androidx.appcompat.widget.AppCompatButton
                android:id="@+id/popup_algorithm_step_img_btn_previous"
                android:layout_width="match_parent"
                android:layout_height="30dp"
                android:layout_gravity="center"
                android:layout_marginStart="@dimen/margin_15dp"
```

```
                android:layout_marginEnd="@dimen/margin_15dp"
                android:layout_weight="1"

                android:background="@drawable/btn_algorithm_previous_step"
                android:text="@string/btn_text_algo_step_back"
                android:textAllCaps="false"
                android:textColor="@color/white" />

            <androidx.appcompat.widget.AppCompatButton
                android:id="@+id/popup_algorithm_step_img_btn_next"
                android:layout_width="match_parent"
                android:layout_height="30dp"
                android:layout_gravity="end"
                android:layout_marginStart="@dimen/margin_15dp"
                android:layout_marginEnd="@dimen/margin_15dp"
                android:layout_weight="1"
                android:background="@drawable/btn_algorithm_next_step"
                android:text="@string/btn_text_algo_step_next"
                android:textAllCaps="false"
                android:textColor="@color/white" />

        </LinearLayout>

        <androidx.appcompat.widget.AppCompatButton
            android:id="@+id/popup_algorithm_step_img_btn_to_end"
            android:layout_width="wrap_content"
            android:layout_height="30dp"
            android:layout_gravity="center"
            android:layout_marginTop="@dimen/margin_15dp"
            android:layout_marginBottom="@dimen/margin_15dp"
            android:background="@drawable/btn_algorithm_to_end"
            android:text="@string/btn_text_algo_step_to_end"
            android:textAllCaps="false"
            android:textColor="@color/white" />


    </LinearLayout>

</layout>
```

fragment_algorithm.kt

```
<?xml version="1.0" encoding="utf-8"?>
<layout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools">

    <data>

        <variable
            name="algorithmViewModel"
            type="com.example.android.bellmanford.algorithm.AlgorithmViewModel"
/>
    </data>

    <androidx.constraintlayout.widget.ConstraintLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:background="@color/white">

        <FrameLayout
            android:id="@+id/fragment_algorithm_flt_canvas"
            android:layout_width="match_parent"
            android:layout_height="match_parent"
            tools:context=".algorithm.AlgorithmFragment">

        </FrameLayout>
```

```
        <ImageButton
            android:id="@+id/btn_back"
            style="@style/BackButton"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_marginTop="@dimen/margin_top_15dp"
            android:contentDescription="@string/content_description_back_button"
            android:onClick="@{() -> algorithmViewModel.onBackNavigate()}"
            app:layout_constraintLeft_toLeftOf="@+id/btn_algo_step"
            app:layout_constraintRight_toRightOf="@id/btn_algo_step"
            app:layout_constraintTop_toTopOf="parent" />

        <ImageButton
            android:id="@+id/btn_algo_step"
            style="@style/AlgorithmStepButton"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_marginTop="@dimen/margin_top_15dp"
                                        android:contentDescription="@string/
content_description_algo_step_button"
            android:onClick="@{() -> algorithmViewModel.onAlgorithmStepShow()}"
            android:visibility="invisible"
            app:layout_constraintLeft_toLeftOf="@+id/btn_algo_info"
            app:layout_constraintRight_toRightOf="@id/btn_algo_info"
            app:layout_constraintTop_toBottomOf="@+id/btn_algo_info"/>

        <ImageButton
            android:id="@+id/btn_algo_info"
            style="@style/AlgorithmInfoButton"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_marginTop="@dimen/margin_15dp"
            app:layout_constraintStart_toStartOf="parent"
            app:layout_constraintTop_toBottomOf="@+id/btn_back"
                                        android:contentDescription="@string/
content_description_algo_info_button"
            android:onClick="@{() -> algorithmViewModel.onAlgorithmInfoShow()}"
             />

        <ImageButton
            android:id="@+id/fragment_algorithm_img_btn_delete_edge"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_marginTop="@dimen/margin_15dp"
            android:layout_marginEnd="@dimen/margin_15dp"
            android:background="@drawable/btn_delete_edge"
            android:visibility="invisible"
            app:layout_constraintEnd_toEndOf="parent"
            app:layout_constraintTop_toTopOf="parent" />

        <ImageButton
            android:id="@+id/fragment_algorithm_img_btn_delete_vertex"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_marginTop="@dimen/margin_15dp"
            android:layout_marginEnd="@dimen/margin_15dp"
            android:background="@drawable/btn_delete_vertex"
            android:visibility="invisible"
            app:layout_constraintEnd_toEndOf="parent"
            app:layout_constraintTop_toTopOf="parent" />

        <ImageButton
            android:id="@+id/fragment_algorithm_img_btn_editing_mode"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_marginEnd="@dimen/margin_15dp"
```

```
            android:layout_marginBottom="@dimen/margin_15dp"
            android:background="@drawable/btn_editing_mode"
            android:visibility="invisible"
            app:layout_constraintBottom_toBottomOf="parent"
            app:layout_constraintEnd_toEndOf="parent"  />

        <ImageButton
            android:id="@+id/fragment_algorithm_img_btn_algorithm_mode"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_marginEnd="@dimen/margin_15dp"
            android:layout_marginBottom="@dimen/margin_15dp"
            android:background="@drawable/btn_algorithm_mode"
            android:visibility="visible"
            app:layout_constraintBottom_toBottomOf="parent"
            app:layout_constraintEnd_toEndOf="parent" />


    </androidx.constraintlayout.widget.ConstraintLayout>


</layout>
```

AppAnimation:

AppAnimation.kt.

```
    package com.example.android.bellmanford.anim

import android.animation.ObjectAnimator
import android.view.View

object AppAnimation {
    fun fadingButtonAnimation(view: View) {
                               ObjectAnimator.ofFloat(view,    View.ALPHA,    0.3F,
1F).setDuration(300).start()
    }
}
```

Экран Developers:

DevelopersFragment.kt.

```
package com.example.android.bellmanford.developers


import android.os.Bundle
import androidx.fragment.app.Fragment
import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import androidx.databinding.DataBindingUtil
import androidx.lifecycle.ViewModelProvider
import androidx.navigation.fragment.findNavController
import com.example.android.bellmanford.R
import com.example.android.bellmanford.anim.AppAnimation
import com.example.android.bellmanford.databinding.FragmentDevelopersBinding
```

```
class DevelopersFragment : Fragment() {

    lateinit var binding: FragmentDevelopersBinding
    lateinit var viewModel: DevelopersViewModel

    override fun onCreateView(
        inflater: LayoutInflater, container: ViewGroup?,
        savedInstanceState: Bundle?
    ): View {

        binding = DataBindingUtil.inflate(
            layoutInflater,
            R.layout.fragment_developers, container, false
        )

        viewModel = ViewModelProvider(this).get(DevelopersViewModel::class.java)

        viewModel.eventBackNavigate.observe(viewLifecycleOwner, { event ->
            if(event) {
                    AppAnimation.fadingButtonAnimation(binding.fragmentDeveloper-
sImgBtnBack)
                findNavController().popBackStack()
                viewModel.onBackNavigateFinish()
            }
        })

        binding.developersViewModel = viewModel

        return binding.root
    }
}
```

DevelopersViewModel.kt.

```
  package com.example.android.bellmanford.developers

import androidx.lifecycle.LiveData
import androidx.lifecycle.MutableLiveData
import androidx.lifecycle.ViewModel

class DevelopersViewModel : ViewModel() {

    private val _eventBackNavigate = MutableLiveData<Boolean>()
    val eventBackNavigate: LiveData<Boolean>
        get() = _eventBackNavigate

    fun onBackNavigate() {
```

```
        _eventBackNavigate.value = true
    }

    fun onBackNavigateFinish() {
        _eventBackNavigate.value = false
    }
}
```

Dialogs:

EdgeWeightDialogFragment.kt.

```
  package com.example.android.bellmanford.dialogs

import android.app.AlertDialog
import android.app.Dialog
import android.os.Build
import android.os.Bundle
import android.view.View
import android.view.WindowInsets
import android.view.WindowInsetsController
import android.widget.EditText
import androidx.fragment.app.DialogFragment
import com.example.android.bellmanford.R
import com.example.android.bellmanford.util.AppFullscreen

class EdgeWeightDialogFragment(val edgeWeightEntered: EdgeWeightEntered) : Di-
alogFragment(){
    override fun onCreateDialog(savedInstanceState: Bundle?): Dialog {
        return activity?.let {


            val builder = AlertDialog.Builder(it)

            val newEdgeWeightDialogView =
                layoutInflater.inflate(R.layout.dialog_edge_weight_picker, null)

            val weightEditText =
                 newEdgeWeightDialogView.findViewById<EditText>(R.id.dialog_ver-
tex_name_picker_et_name)

            builder.setView(newEdgeWeightDialogView)
                .setTitle(getString(R.string.title_dialog_edge_weight_picker))
                .setPositiveButton(
                    getString(R.string.txt_dialog_positive_button)
                ) { _, _ ->
                            edgeWeightEntered.receiveWeight(weightEditText.text.-
toString())
                    dialog?.cancel()
                }
                .setNegativeButton(
                    getString(R.string.txt_dialog_negative_button)
                ) { _, _ ->
                    dialog?.cancel()
                }
            builder.create()
        } ?: throw IllegalStateException("Activity cannot be null")

    }

    override fun onDetach() {
        super.onDetach()
        AppFullscreen.turnFullscreen(requireActivity())
        edgeWeightEntered.dialogClosed()
```

```
    }
}

interface EdgeWeightEntered {
    fun receiveWeight(weight: String)
    fun dialogClosed()
}
```

VertexNameDialogFragment.kt.

```
package com.example.android.bellmanford.dialogs

import android.app.AlertDialog
import android.app.Dialog
import android.os.Build
import android.os.Bundle
import android.view.View
import android.view.WindowInsets
import android.view.WindowInsetsController
import android.widget.EditText
import android.widget.Toast
import androidx.annotation.StringRes
import androidx.fragment.app.DialogFragment
import com.example.android.bellmanford.R
import com.example.android.bellmanford.util.AppFullscreen

class VertexNameDialogFragment(val nameEntered: VertexNameEntered) : DialogFrag-
ment() {


    override fun onCreateDialog(savedInstanceState: Bundle?): Dialog {
        return activity?.let {


            val builder = AlertDialog.Builder(it)

            val newVertexNameDialogView =
                layoutInflater.inflate(R.layout.dialog_vertex_name_picker, null)

            val nameEditText =
                 newVertexNameDialogView.findViewById<EditText>(R.id.dialog_ver-
tex_name_picker_et_name)

            builder.setView(newVertexNameDialogView)
                .setTitle(getString(R.string.title_dialog_vertex_name_picker))
                .setPositiveButton(
                    getString(R.string.txt_dialog_positive_button)
                ) { _, _ ->
```

```
                    nameEntered.receiveName(nameEditText.text.toString())
                    dialog?.cancel()
                }
                .setNegativeButton(
                    getString(R.string.txt_dialog_negative_button)
                ) { _, _ ->
                    dialog?.cancel()
                }
            builder.create()
        } ?: throw IllegalStateException("Activity cannot be null")
    }

    override fun onDetach() {
        super.onDetach()
        AppFullscreen.turnFullscreen(requireActivity())
    }
}

interface VertexNameEntered {
    fun receiveName(name: String)
}
```

Экран AlgorithmInfo:

AlgorithmInfoFragment.kt.

```
package com.example.android.bellmanford.info

import android.os.Bundle
import androidx.fragment.app.Fragment
import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import androidx.databinding.DataBindingUtil
import androidx.lifecycle.ViewModelProvider
import androidx.navigation.fragment.findNavController
import com.example.android.bellmanford.R
import com.example.android.bellmanford.anim.AppAnimation
import com.example.android.bellmanford.databinding.FragmentAlgorithmInfoBinding

class AlgorithmInfoFragment : Fragment() {

    lateinit var binding: FragmentAlgorithmInfoBinding
    lateinit var viewModel: AlgorithmInfoViewModel

    override fun onCreateView(
        inflater: LayoutInflater, container: ViewGroup?,
        savedInstanceState: Bundle?
    ): View {

        binding = DataBindingUtil.inflate(
            layoutInflater,
            R.layout.fragment_algorithm_info, container, false
```

```
        )

         viewModel = ViewModelProvider(this).get(AlgorithmInfoViewModel::class.-
java)

        viewModel.eventBackNavigate.observe(viewLifecycleOwner, { event ->
            if(event) {
                 AppAnimation.fadingButtonAnimation(binding.fragmentAlgorithmIn-
foImgBtnBack)
                findNavController().popBackStack()
                viewModel.onBackNavigateFinish()
            }
        })

        binding.infoViewModel = viewModel

        return binding.root
    }

}
```

AlgorithmInfoViewModel.kt.

```
   package com.example.android.bellmanford.info

import androidx.lifecycle.LiveData
import androidx.lifecycle.MutableLiveData
import androidx.lifecycle.ViewModel

class AlgorithmInfoViewModel : ViewModel() {

    private val _eventBackNavigate = MutableLiveData<Boolean>()
    val eventBackNavigate: LiveData<Boolean>
        get() = _eventBackNavigate

    fun onBackNavigate() {
        _eventBackNavigate.value = true
    }

    fun onBackNavigateFinish() {
        _eventBackNavigate.value = false
    }
}
```

Экран Start:

StartFragment.kt.

```
package com.example.android.bellmanford.start

import android.os.Bundle
import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import androidx.databinding.DataBindingUtil
import androidx.fragment.app.Fragment
import androidx.lifecycle.ViewModelProvider
import androidx.navigation.fragment.findNavController
import com.example.android.bellmanford.R
import com.example.android.bellmanford.anim.AppAnimation
import com.example.android.bellmanford.databinding.FragmentStartBinding
```

```
class StartFragment : Fragment() {

    lateinit var binding: FragmentStartBinding
    lateinit var viewModel: StartViewModel

    override fun onCreateView(
        inflater: LayoutInflater, container: ViewGroup?,
        savedInstanceState: Bundle?
    ): View {

        binding = DataBindingUtil.inflate(
            layoutInflater,
            R.layout.fragment_start, container, false
        )

        viewModel = ViewModelProvider(this).get(StartViewModel::class.java)

        binding.startViewModel = viewModel

        viewModel.eventInfoNavigate.observe(viewLifecycleOwner, { event ->
            if (event) {
                 findNavController().navigate(R.id.action_startFragment_to_algo-
rithmInfoFragment)
                AppAnimation.fadingButtonAnimation(binding.fragmentStartBtnAlgo-
rithmInfo)
                viewModel.onInfoNavigateFinish ()
            }
        })

        viewModel.eventDevelopersNavigate.observe(viewLifecycleOwner, { event ->
            if (event) {
                findNavController().navigate(R.id.action_startFragment_to_devel-
opersFragment)
                  AppAnimation.fadingButtonAnimation(binding.fragmentStartBtnDe-
velopers)
                viewModel.onDevelopersNavigateFinish()
            }
        })

        viewModel.eventAlgorithmNavigate.observe(viewLifecycleOwner, { event ->
            if (event) {
                 findNavController().navigate(R.id.action_startFragment_to_algo-
rithmFragment)
                    AppAnimation.fadingButtonAnimation(binding.fragmentStartBt-
nTryAlgorithm)
                viewModel.onAlgorithmNavigateFinish()
            }
        })

        viewModel.eventCloseApp.observe(viewLifecycleOwner, { event->
            if(event) {
                  AppAnimation.fadingButtonAnimation(binding.fragmentStartImgBt-
nCloseApp)
                requireActivity().finish()
            }
        })

        return binding.root
    }

}
```

StartViewModel.kt.

```
package com.example.android.bellmanford.start

import androidx.lifecycle.LiveData
import androidx.lifecycle.MutableLiveData
import androidx.lifecycle.ViewModel

class StartViewModel : ViewModel() {

    private val _eventInfoNavigate = MutableLiveData<Boolean>()
    val eventInfoNavigate: LiveData<Boolean>
        get() = _eventInfoNavigate

    fun onInfoNavigate() {
        _eventInfoNavigate.value = true
    }

    fun onInfoNavigateFinish() {
        _eventInfoNavigate.value = false
    }

    private val _eventDevelopersNavigate = MutableLiveData<Boolean>()
    val eventDevelopersNavigate: LiveData<Boolean>
        get() = _eventDevelopersNavigate

    fun onDevelopersNavigate() {
        _eventDevelopersNavigate.value = true
    }

    fun onDevelopersNavigateFinish() {
        _eventDevelopersNavigate.value = false
    }

    private val _eventAlgorithmNavigate = MutableLiveData<Boolean>()
    val eventAlgorithmNavigate: LiveData<Boolean>
        get() = _eventAlgorithmNavigate

    fun onAlgorithmNavigate() {
        _eventAlgorithmNavigate.value = true
    }

    fun onAlgorithmNavigateFinish() {
        _eventAlgorithmNavigate.value = false
    }

    private val _eventCloseApp = MutableLiveData<Boolean>()
    val eventCloseApp: LiveData<Boolean>
        get() = _eventCloseApp

    fun onCloseApp() {
        _eventCloseApp.value = true
    }
}
```

Util:

AppFullscreen.kt.

```
package com.example.android.bellmanford.util

import android.app.Activity
import android.os.Build
import android.view.View
import android.view.WindowInsets
import android.view.WindowInsetsController
```

```
object AppFullscreen {
    fun turnFullscreen(activity: Activity) {
        if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.R) {
            activity.window.setDecorFitsSystemWindows(false)
            activity.window.insetsController?.let {
                it.hide(WindowInsets.Type.statusBars() or WindowInsets.Type.nav-
igationBars())
                                                        it.systemBarsBehavior   =
WindowInsetsController.BEHAVIOR_SHOW_TRANSIENT_BARS_BY_SWIPE
            }
        } else {
            @Suppress("DEPRECATION")
                                    activity.window.decorView.systemUiVisibility   =
(View.SYSTEM_UI_FLAG_FULLSCREEN
                    or View.SYSTEM_UI_FLAG_HIDE_NAVIGATION
                    or View.SYSTEM_UI_FLAG_IMMERSIVE_STICKY
                    or View.SYSTEM_UI_FLAG_LAYOUT_STABLE
                    or View.SYSTEM_UI_FLAG_LAYOUT_FULLSCREEN
                    or View.SYSTEM_UI_FLAG_LAYOUT_HIDE_NAVIGATION)
        }
    }
}
```

MainActivity.kt.

```
package com.example.android.bellmanford

import android.os.Build
import androidx.appcompat.app.AppCompatActivity
import android.os.Bundle
import android.view.View
import android.view.WindowInsets
import android.view.WindowInsetsController
import com.example.android.bellmanford.util.AppFullscreen

class MainActivity : AppCompatActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)
        AppFullscreen.turnFullscreen(this)
    }

    override fun onRestart() {
        super.onRestart()
        AppFullscreen.turnFullscreen(this)
    }
}
```

Ресурсы:

nav_graph.xml.

```
<?xml version="1.0" encoding="utf-8"?>
<navigation xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/nav_graph"
    app:startDestination="@id/startFragment">
    <fragment
        android:id="@+id/startFragment"
        android:name="com.example.android.bellmanford.start.StartFragment"
        android:label="StartFragment" >
        <action
            android:id="@+id/action_startFragment_to_developersFragment"
            app:enterAnim="@anim/slide_in_left"
            app:exitAnim="@anim/wait_anim"
            app:popEnterAnim="@anim/wait_anim"
            app:popExitAnim="@anim/slide_in_right"
            app:destination="@id/developersFragment" />
        <action
            android:id="@+id/action_startFragment_to_algorithmInfoFragment"
            app:enterAnim="@anim/slide_in_left"
            app:exitAnim="@anim/wait_anim"
            app:popEnterAnim="@anim/wait_anim"
            app:popExitAnim="@anim/slide_in_right"
            app:destination="@id/algorithmInfoFragment" />
        <action
            android:id="@+id/action_startFragment_to_algorithmFragment"
            app:enterAnim="@anim/slide_in_left"
            app:exitAnim="@anim/wait_anim"
            app:popEnterAnim="@anim/wait_anim"
            app:popExitAnim="@anim/slide_in_right"
            app:destination="@id/algorithmFragment" />
    </fragment>
    <fragment
        android:id="@+id/developersFragment"
        android:name="com.example.android.bellmanford.developers.DevelopersFrag-
ment"
        android:label="DevelopersFragment" />
    <fragment
        android:id="@+id/algorithmInfoFragment"
           android:name="com.example.android.bellmanford.info.AlgorithmInfoFrag-
ment"
        android:label="AlgorithmInfoFragment" />
    <fragment
        android:id="@+id/algorithmFragment"
```

```
          android:name="com.example.android.bellmanford.algorithm.AlgorithmFrag-
ment"
        android:label="AlgorithmFragment" />
</navigation>
```

activity_main.xml

```
  <?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout        xmlns:android="http://
schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <androidx.fragment.app.FragmentContainerView
        android:id="@+id/nav_host_fragment"
        android:name="androidx.navigation.fragment.NavHostFragment"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        app:defaultNavHost="true"
        app:navGraph="@navigation/nav_graph" />

</androidx.constraintlayout.widget.ConstraintLayout>
```

algoinfo_popup.xml

```
 <?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout        xmlns:android="http://
schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content">

    <LinearLayout
        android:id="@+id/linearLayout2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:background="@color/background_grey"
        android:gravity="center"
        android:orientation="vertical"
        android:paddingLeft="10dp"
        android:paddingRight="10dp"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent">

        <LinearLayout
            android:layout_width="wrap_content"
            android:layout_height="250dp"
            android:orientation="vertical"
            android:padding="10dp">

            <pl.droidsonroids.gif.GifImageView
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:src="@drawable/infogif" />
```

```
        </LinearLayout>
    </LinearLayout>

    <ImageButton
        android:id="@+id/exit_button"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:background="@drawable/btn_popup_exit"

        app:layout_constraintEnd_toEndOf="@id/linearLayout2"
        app:layout_constraintTop_toTopOf="parent">


    </ImageButton>
</androidx.constraintlayout.widget.ConstraintLayout>
```

algostep_popup.xml

```
<?xml version="1.0" encoding="utf-8"?>
<layout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto">

    <LinearLayout
        android:layout_width="wrap_content"
        android:layout_height="match_parent"
        android:background="@color/background_grey"
        android:orientation="vertical">

        <ImageButton
            android:id="@+id/popup_algorithm_step_img_btn_close"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_gravity="end"
            android:layout_marginTop="@dimen/margin_15dp"
            android:layout_marginEnd="@dimen/margin_15dp"
            android:layout_marginBottom="@dimen/margin_15dp"
            android:background="@drawable/btn_popup_exit" />


        <androidx.recyclerview.widget.RecyclerView
            android:id="@+id/popup_algorithm_step_rv_step_explanation"
            android:layout_width="300dp"
            android:layout_height="0dp"
            android:layout_marginStart="5dp"
            android:layout_marginEnd="5dp"
            android:layout_weight="1"
            android:background="@drawable/view_rv_background"
            app:layoutManager="androidx.recyclerview.widget.LinearLayoutManager"
/>

        <LinearLayout
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_gravity="center"
            android:layout_marginTop="@dimen/margin_15dp"
            android:orientation="horizontal">

            <androidx.appcompat.widget.AppCompatButton
                android:id="@+id/popup_algorithm_step_img_btn_previous"
                android:layout_width="match_parent"
                android:layout_height="30dp"
                android:layout_gravity="center"
                android:layout_marginStart="@dimen/margin_15dp"
                android:layout_marginEnd="@dimen/margin_15dp"
```

```
                android:layout_weight="1"

                android:background="@drawable/btn_algorithm_previous_step"
                android:text="@string/btn_text_algo_step_back"
                android:textAllCaps="false"
                android:textColor="@color/white" />

            <androidx.appcompat.widget.AppCompatButton
                android:id="@+id/popup_algorithm_step_img_btn_next"
                android:layout_width="match_parent"
                android:layout_height="30dp"
                android:layout_gravity="end"
                android:layout_marginStart="@dimen/margin_15dp"
                android:layout_marginEnd="@dimen/margin_15dp"
                android:layout_weight="1"
                android:background="@drawable/btn_algorithm_next_step"
                android:text="@string/btn_text_algo_step_next"
                android:textAllCaps="false"
                android:textColor="@color/white" />

        </LinearLayout>

        <androidx.appcompat.widget.AppCompatButton
            android:id="@+id/popup_algorithm_step_img_btn_to_end"
            android:layout_width="wrap_content"
            android:layout_height="30dp"
            android:layout_gravity="center"
            android:layout_marginTop="@dimen/margin_15dp"
            android:layout_marginBottom="@dimen/margin_15dp"
            android:background="@drawable/btn_algorithm_to_end"
            android:text="@string/btn_text_algo_step_to_end"
            android:textAllCaps="false"
            android:textColor="@color/white" />


    </LinearLayout>

</layout>
```

dialog_edge_weight_picker.xml

```
<?xml version="1.0" encoding="utf-8"?>
<layout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools">
    <FrameLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent">

        <EditText
            android:id="@+id/dialog_vertex_name_picker_et_name"
            android:layout_marginStart="@dimen/margin_dialog_16dp"
            android:layout_marginEnd="@dimen/margin_dialog_16dp"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:maxLength="3"
            android:ems="10"
            android:inputType="numberSigned"
            android:hint="@string/hint_edge_weigt_input" />
    </FrameLayout>
</layout>
```

dialog_vertex_name_picker.xml

```
<?xml version="1.0" encoding="utf-8"?>
<layout>
    <FrameLayout
        xmlns:android="http://schemas.android.com/apk/res/android"
        xmlns:tools="http://schemas.android.com/tools"
        android:layout_width="match_parent"
        android:layout_height="match_parent">

        <EditText
            android:id="@+id/dialog_vertex_name_picker_et_name"
            android:layout_marginStart="@dimen/margin_dialog_16dp"
            android:layout_marginEnd="@dimen/margin_dialog_16dp"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:hint="@string/hint_vertex_name_input" />
    </FrameLayout>
</layout>
```

fragment_algortihm.xml

```
<?xml version="1.0" encoding="utf-8"?>
<layout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools">

    <data>

        <variable
            name="algorithmViewModel"
            type="com.example.android.bellmanford.algorithm.AlgorithmViewModel"
/>
    </data>

    <androidx.constraintlayout.widget.ConstraintLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:background="@color/white">

        <FrameLayout
            android:id="@+id/fragment_algorithm_flt_canvas"
            android:layout_width="match_parent"
            android:layout_height="match_parent"
            tools:context=".algorithm.AlgorithmFragment">

        </FrameLayout>

        <ImageButton
            android:id="@+id/btn_back"
            style="@style/BackButton"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_marginTop="@dimen/margin_top_15dp"
            android:contentDescription="@string/content_description_back_button"
            android:onClick="@{() -> algorithmViewModel.onBackNavigate()}"
            app:layout_constraintLeft_toLeftOf="@+id/btn_algo_step"
            app:layout_constraintRight_toRightOf="@id/btn_algo_step"
            app:layout_constraintTop_toTopOf="parent" />

        <ImageButton
            android:id="@+id/btn_algo_step"
            style="@style/AlgorithmStepButton"
            android:layout_width="wrap_content"
```

```
            android:layout_height="wrap_content"
            android:layout_marginTop="@dimen/margin_top_15dp"
                                            android:contentDescription="@string/
content_description_algo_step_button"
            android:onClick="@{() -> algorithmViewModel.onAlgorithmStepShow()}"
            android:visibility="invisible"
            app:layout_constraintLeft_toLeftOf="@+id/btn_algo_info"
            app:layout_constraintRight_toRightOf="@id/btn_algo_info"
            app:layout_constraintTop_toBottomOf="@+id/btn_algo_info"/>

        <ImageButton
            android:id="@+id/btn_algo_info"
            style="@style/AlgorithmInfoButton"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_marginTop="@dimen/margin_15dp"
            app:layout_constraintStart_toStartOf="parent"
            app:layout_constraintTop_toBottomOf="@+id/btn_back"
                                            android:contentDescription="@string/
content_description_algo_info_button"
            android:onClick="@{() -> algorithmViewModel.onAlgorithmInfoShow()}"
             />

        <ImageButton
            android:id="@+id/fragment_algorithm_img_btn_delete_edge"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_marginTop="@dimen/margin_15dp"
            android:layout_marginEnd="@dimen/margin_15dp"
            android:background="@drawable/btn_delete_edge"
            android:visibility="invisible"
            app:layout_constraintEnd_toEndOf="parent"
            app:layout_constraintTop_toTopOf="parent" />

        <ImageButton
            android:id="@+id/fragment_algorithm_img_btn_delete_vertex"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_marginTop="@dimen/margin_15dp"
            android:layout_marginEnd="@dimen/margin_15dp"
            android:background="@drawable/btn_delete_vertex"
            android:visibility="invisible"
            app:layout_constraintEnd_toEndOf="parent"
            app:layout_constraintTop_toTopOf="parent" />

        <ImageButton
            android:id="@+id/fragment_algorithm_img_btn_editing_mode"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_marginEnd="@dimen/margin_15dp"
            android:layout_marginBottom="@dimen/margin_15dp"
            android:background="@drawable/btn_editing_mode"
            android:visibility="invisible"
            app:layout_constraintBottom_toBottomOf="parent"
            app:layout_constraintEnd_toEndOf="parent"  />

        <ImageButton
            android:id="@+id/fragment_algorithm_img_btn_algorithm_mode"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_marginEnd="@dimen/margin_15dp"
            android:layout_marginBottom="@dimen/margin_15dp"
            android:background="@drawable/btn_algorithm_mode"
            android:visibility="visible"
            app:layout_constraintBottom_toBottomOf="parent"
            app:layout_constraintEnd_toEndOf="parent" />
```

```
    </androidx.constraintlayout.widget.ConstraintLayout>


</layout>
```

fragment_algorithm_info.xml

```
<?xml version="1.0" encoding="utf-8"?>
<layout xmlns:android="http://schemas.android.com/apk/res/android">

    <data>

        <variable
            name="infoViewModel"
            type="com.example.android.bellmanford.info.AlgorithmInfoViewModel" /
>
    </data>

    <ScrollView
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:background="@color/background_grey">


        <LinearLayout
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
                                            android:layout_marginStart="@dimen/
margin_fragment_info_lt_start_end"
            android:layout_marginEnd="@dimen/margin_fragment_info_lt_start_end"
            android:orientation="vertical">

            <ImageButton
                android:id="@+id/fragment_algorithm_info_img_btn_back"
                style="@style/BackButton"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:layout_marginTop="@dimen/margin_top_15dp"
                                            android:contentDescription="@string/
content_description_back_button"
                android:onClick="@{() -> infoViewModel.onBackNavigate()}" />

            <TextView
                style="@style/TextAppearance.Header"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:text="@string/txt_algorithm_name" />

            <TextView
                style="@style/TextAppearance.Default"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:text="Hello world!" />

        </LinearLayout>
    </ScrollView>
</layout>
```

fragment_developers.xml

```
<?xml version="1.0" encoding="utf-8"?>
<layout xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:android="http://schemas.android.com/apk/res/android">

    <data>
        <variable
            name="developersViewModel"
                type="com.example.android.bellmanford.developers.DevelopersView-
Model" />
    </data>

    <ScrollView

        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:background="@color/background_grey">

        <LinearLayout
                                                android:layout_marginStart="@dimen/
margin_fragment_developers_lt_start_end"
                                                  android:layout_marginEnd="@dimen/
margin_fragment_developers_lt_start_end"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:orientation="vertical">

            <ImageButton
                android:id="@+id/fragment_developers_img_btn_back"
                style="@style/BackButton"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:layout_marginTop="@dimen/margin_top_15dp"
                                                android:contentDescription="@string/
content_description_back_button"
                android:onClick="@{() -> developersViewModel.onBackNavigate()}"/
>

            <TextView
                style="@style/TextAppearance.Header"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:text="@string/txt_developers_header" />

            <TextView
                style="@style/TextAppearance.Default"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:text="@string/txt_developers_info"/>

        </LinearLayout>
    </ScrollView>


</layout>
```

fragment_start.xml

```
<?xml version="1.0" encoding="utf-8"?>

<layout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools">

    <data>
```

```
        <variable
            name="startViewModel"
            type="com.example.android.bellmanford.start.StartViewModel" />
    </data>

    <androidx.constraintlayout.widget.ConstraintLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:background="@color/background_grey"
        tools:context=".start.StartFragment">

        <LinearLayout
            android:id="@+id/fragment_start_llt_buttons_container"
            android:layout_width="300dp"
            android:layout_height="wrap_content"
            android:layout_marginStart="15dp"
            android:orientation="vertical"
            app:layout_constraintStart_toStartOf="parent"
            app:layout_constraintTop_toTopOf="parent">

            <androidx.appcompat.widget.AppCompatButton
                android:id="@+id/fragment_start_btn_try_algorithm"
                style="@style/StartFragmentButton"
                android:layout_width="match_parent"
                android:layout_height="45dp"
                android:layout_marginTop="15dp"
                android:onClick="@{() -> startViewModel.onAlgorithmNavigate()}"
                android:text="@string/btn_text_try_algorithm" />

            <androidx.appcompat.widget.AppCompatButton
                android:id="@+id/fragment_start_btn_algorithm_info"
                style="@style/StartFragmentButton"
                android:layout_width="match_parent"
                android:layout_height="45dp"
                android:layout_marginTop="15dp"
                android:onClick="@{() -> startViewModel.onInfoNavigate()}"
                android:text="@string/btn_text_algorithm_info" />

            <androidx.appcompat.widget.AppCompatButton
                android:id="@+id/fragment_start_btn_developers"
                style="@style/StartFragmentButton"
                android:layout_width="match_parent"
                android:layout_height="45dp"
                android:layout_marginTop="15dp"
                android:onClick="@{() -> startViewModel.onDevelopersNavigate()}"
                android:text="@string/btn_text_developers" />

        </LinearLayout>

        <ImageView
            android:layout_width="0dp"
            android:layout_height="0dp"
            android:adjustViewBounds="true"
            android:importantForAccessibility="no"
            android:paddingStart="@dimen/padding_fragment_start_logo"
            android:paddingTop="@dimen/padding_fragment_start_logo"
            android:paddingEnd="@dimen/padding_fragment_start_logo"
            android:paddingBottom="@dimen/padding_fragment_start_logo"
            android:src="@drawable/ic_start_screen_logo"
            app:layout_constraintBottom_toBottomOf="parent"
            app:layout_constraintEnd_toEndOf="parent"
            app:layout_constraintHorizontal_bias="1.0"
                                        app:layout_constraintStart_toEndOf="@+id/
fragment_start_llt_buttons_container"
            app:layout_constraintTop_toTopOf="parent" />
```

```
        <ImageButton
            android:id="@+id/fragment_start_img_btn_close_app"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_marginBottom="@dimen/margin_top_15dp"
            android:layout_marginStart="@dimen/margin_15dp"
            android:contentDescription="@string/content_description_back_button"
            android:background="@drawable/btn_exit_app"
            android:onClick="@{() -> startViewModel.onCloseApp()}"
            app:layout_constraintBottom_toBottomOf="parent"
            app:layout_constraintStart_toStartOf="parent"/>

    </androidx.constraintlayout.widget.ConstraintLayout>


</layout>
```

text_item_view.xml

```
<?xml version="1.0" encoding="utf-8"?>
<TextView xmlns:android="http://schemas.android.com/apk/res/android"
    android:textSize="12sp"
    android:paddingStart="@dimen/margin_15dp"
    android:paddingEnd="@dimen/margin_15dp"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
android:textColor="@android:color/black"/>
```