

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

ОТЧЕТ
по учебной практике
ТЕМА: «АХО-КОРАСИК»

Студент гр. 1303

Голов О.С.

Студент гр. 1303

Токун Г.С.

Руководитель

Фирсов М.А.

Санкт-Петербург

2023

ЗАДАНИЕ НА УЧЕБНУЮ ПРАКТИКУ

Студент Голов О.С. группы 1303

Студент Тоkun Г.С. группы 1303

Тема практики: Ахо-Корасик

Задание на практику:

Командная итеративная разработка визуализатора алгоритма на Kotlin с графическим интерфейсом.

Алгоритм: Поиск подстрок в строке Ахо-Корасик

Сроки прохождения практики: 30.06.2023 – 13.07.2023

Дата сдачи отчета: 13.07.2023

Дата защиты отчета: 13.07.2023

Студент гр. 1303

Голов О.С.

Студент гр. 1303

Тоkun Г.С.

Руководитель

Фирсов М.А.

АННОТАЦИЯ

Целью данного проекта было создание визуализации с возможной интерактивностью алгоритма Ахо-Корасик, который используется для поиска нескольких подстрок в строке. Работа написана на языке программирования Kotlin. Упор делался на возможности манипулировать бором, который является основой алгоритма.

Выполнение работы включало в себя такие итерации как: создание спецификации и плана тестирования; написание кода, реализующего алгоритм; создание прототипа; визуализация алгоритма; написание отчета. Программа на вход принимает два набора текста вводимых через поля для ввода внутри приложения и выводит изображения бора, конечного автомата и вхождения подстрок в текст. После этого у пользователя имеется возможность добавить вершину в бор или поменять терминальное состояние любой из них.

SUMMARY

The goal of this project was to create a visualization with possible interactivity of the Aho-Korasik algorithm, which is used to find multiple substrings in a string. The work was written in the Kotlin programming language. Emphasis was placed on the ability to manipulate the trie, which is the basis of the algorithm.

The execution of the work involved iterations such as: creating a specification and test plan; writing code that implements the algorithm; creating a prototype; visualizing the algorithm; and writing a report. The program takes as input two sets of text entered through input fields within the application and outputs images of the trie, finite automaton and occurrences of substrings in the text. The user then has the option of adding a vertex to the trie or changing the terminal state of any of them.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ.....	5
1. ТРЕБОВАНИЯ К ПРОГРАММЕ.....	6
1.1. Исходные Требования к программе.....	6
1.1.1. Формальная постановка задачи.....	6
1.1.2. Описание интерфейса.....	6
1.1.3. Формат входных и выходных данных.....	8
1.2. Уточнения требований после сдачи 1-ой версии.....	9
1.3. Уточнения требований после сдачи 2-ой версии.....	9
2. ПЛАН РАЗРАБОТКИ И РАСПРЕДЕЛЕНИЕ РОЛЕЙ В БРИГАДЕ....	10
2.1. План разработки.....	10
2.2. Распределение ролей в бригаде.....	10
3. ОСОБЕННОСТИ РЕАЛИЗАЦИИ.....	12
3.1. Структуры данных.....	12
3.2. Основные методы.....	14
4. ТЕСТИРОВАНИЕ.....	16
4.1. Тестирование граничных условий.....	20
4.2. Тестирование интерфейса.....	22
4.3. Тестирование структуры данных.....	26
ЗАКЛЮЧЕНИЕ.....	28
СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ.....	29

ВВЕДЕНИЕ

Целью данного проекта было создание интерактивного приложения по визуализации алгоритма Ахо-корасик

Задачи проекта:

- Реализация алгоритма Ахо-Корасик.
- Визуализация бора: Вывод дерева и конечного автомата.
- Добавление пользовательского интерфейса: имплементация возможности нажатием кнопок добавить в бор новую вершину или сменить терминальность

1. ТРЕБОВАНИЯ К ПРОГРАММЕ

1.1. Исходные Требования к программе

1.1.1. Формальная постановка задачи

Задачей проекта является создание интерактивного приложения на языке Kotlin, которое отображает работу алгоритма Ахо-Корасик и позволяет взаимодействовать с ним, используя графический интерфейс.

1.1.2. Работа программы

1) Приложение открывает окно с двумя полями для ввода текста, в которые пользователь вводит текст в первое поле и набор шаблонов во второе поле и по нажатию клавиши Enter или кнопки на интерфейсе “отправить” программа считывает данные(рис. 2).

2) Строит по ним бор в терминах алгоритма Ахо-Корасик, после чего выводит на экран изображение бора слева и конечного автомата справа(рис. 3).

3) Через дополнительное поле вводятся данные для постройки нового состояния по форме(“название состояния”->”символ перехода”)(пример текста для добавления ребра: abc->c)(рис.4) и по нажатию клавиши Enter или кнопки на интерфейсе “отправить” программа считывает данные, после чего в бор добавляется ребро из первого состояния в новое, автомат перестраивается по новому бору, приложение анализирует изменённый бор и по нему собирает набор шаблонов в зависимости от данных введённых пользователем(рис. 5). В случае если переход по символу уже имеется то ничего не произойдёт.

4) Через второе дополнительное окно вводится название состояния для которого нужно изменить статус с конечного на промежуточное и наоборот, после ввода и нажатия клавиши Enter или кнопки на интерфейсе “отправить”, бор изменится, а автомат перестроится

1.1.3. Описание интерфейса

При запуске приложение имеет два поля для ввода текста(рис. 1), в одно вписывается текст, в другое набор шаблонов разделённых символом “#”(пример набора шаблонов: #ab#ba#aba) на каждом поле имеется интерфейсная кнопка “отправить” по которой пользователь подтверждает введённые данные(рис. 2).

После подтверждения данных на экран выводится два графа представляющих собой бор и конечный автомат, который и является графическим отображением бора с выводом всех вхождений шаблонов в текст по принципу ("номер шаблона" "позиция вхождения") промежуточные значения имеют черную обводку, конечные значения имеют красную обводку, из вершин исходят направленные рёбра. Рёбра зелёного цвета

служат обозначением конечных ссылок, а рёбра синего цвета -- суффиксных ссылок (рис. 3).

В окне с графом также имеются два поля для ввода текста, благодаря одному можно добавлять рёбра по форме(“название состояния”->”символ перехода”)(рис. 4 и 5), по второму можно переключать статус состояния с конечного на промежуточное. Для подтверждения данных введённых пользователем можно нажать кнопку Enter, так же в каждом поле имеется кнопка “отправить” (рис. 6 и 7)

Step 1 начало работы

Введите текст

Введите шаблоны через #

Рис. 1 – примерная иллюстрация открывшегося приложения

Step 2 ввод данных

abab
отправить

#a#ba
отправить

Рис. 2 – ввод данных для построения бора и нахождения подстрок в строке

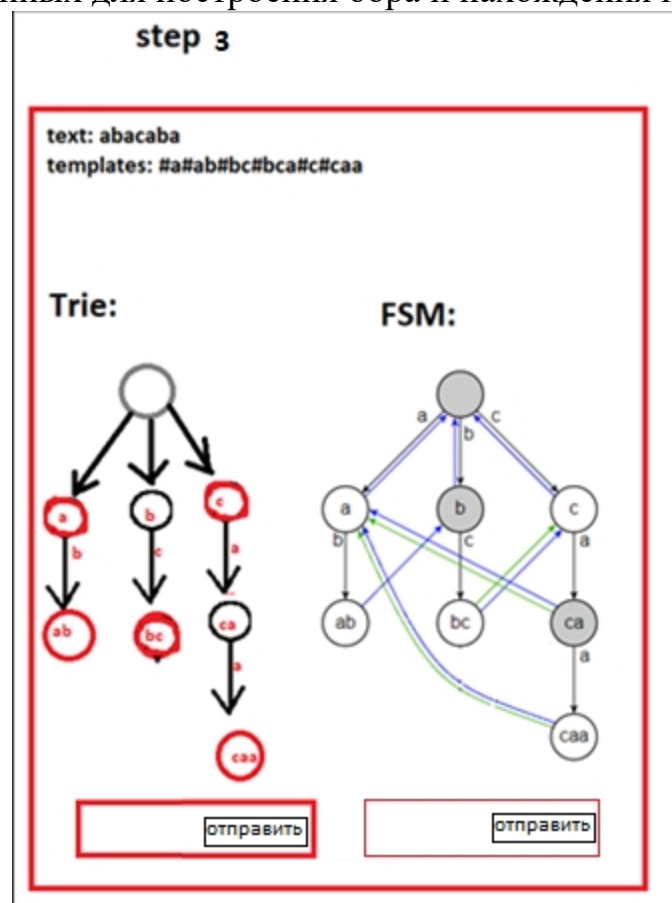


Рис. 3 – иллюстрация вывода

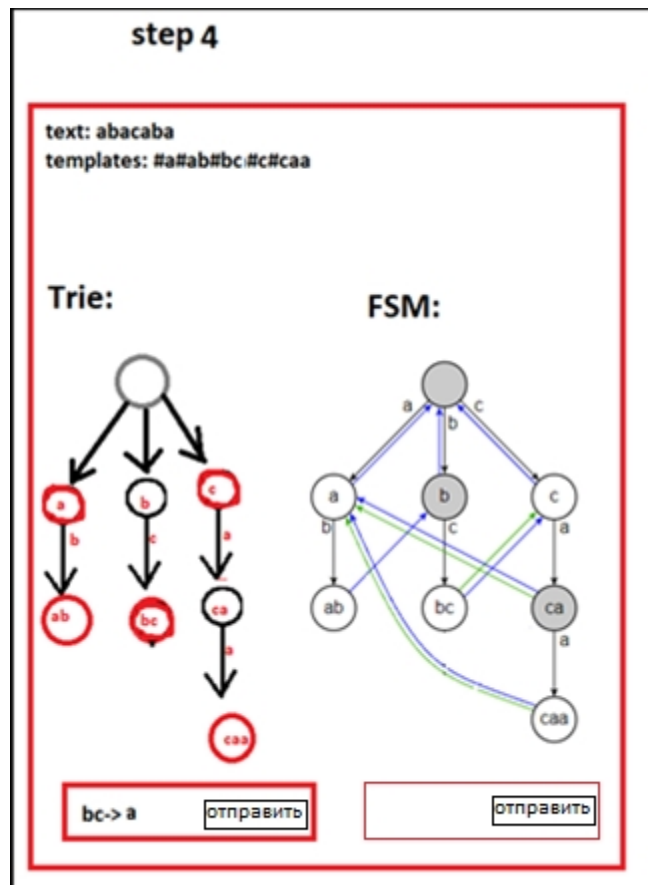


Рис. 4 – иллюстрация записи добавления вершины в бор

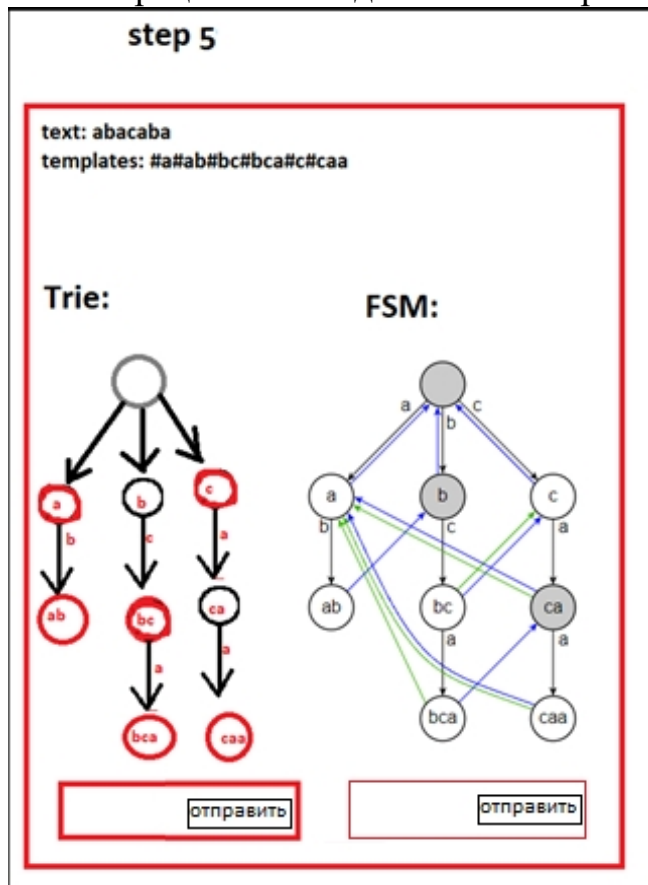


Рис. 5 – иллюстрация изменённого автомата и бора после добавления ребра

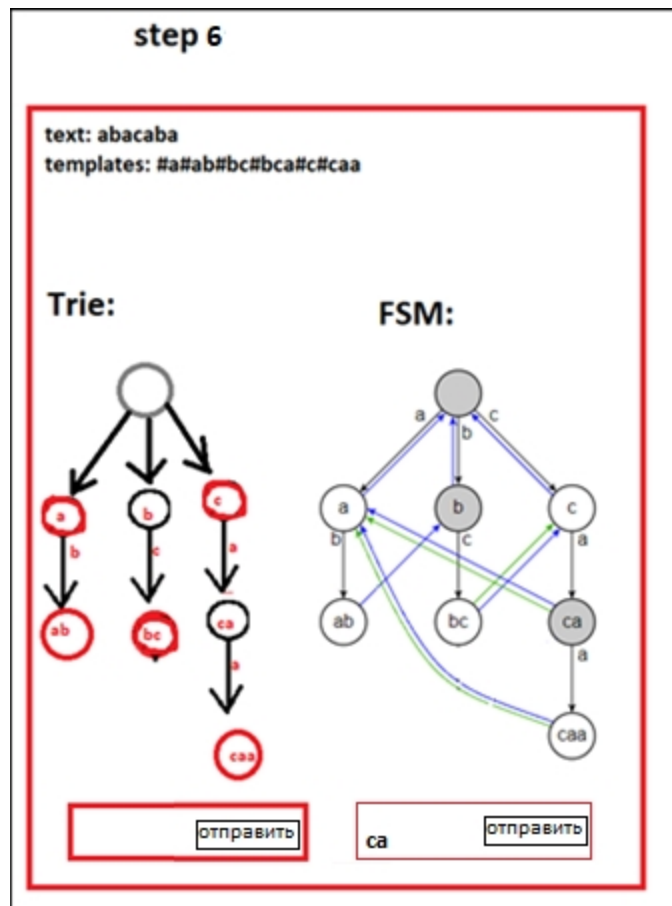


Рис. 6 – иллюстрация записи изменения статуса состояния

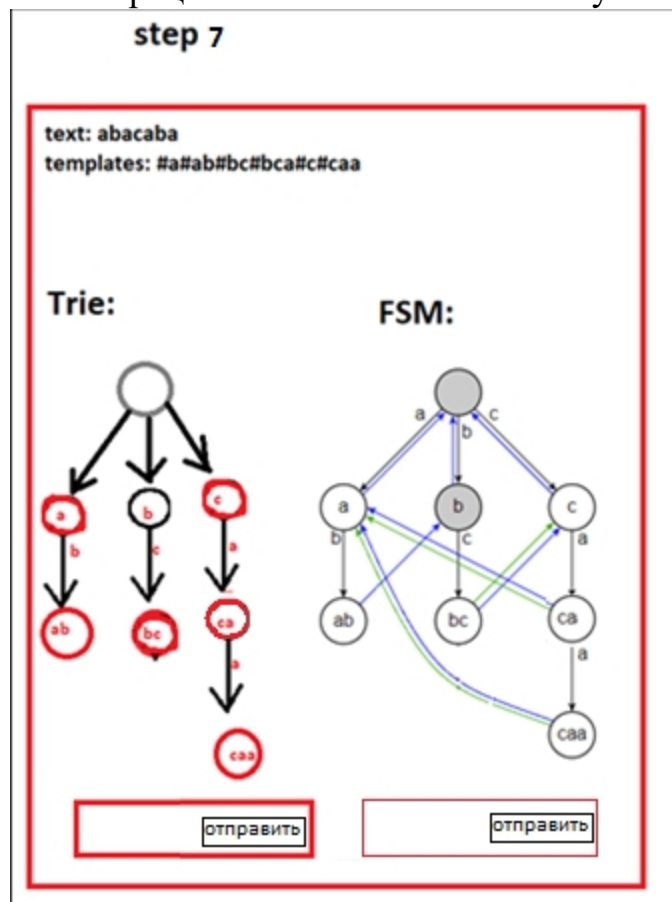


Рис. 7 – иллюстрация изменения бора и автомата

1.1.4. Формат входных и выходных данных

Вход: строка представляющая собой текст в первом окне, серия строк разделённых разделительным знаком “#” во втором окне.

Вывод: Все вхождения шаблонов в текст по принципу ("номер шаблона" "позиция вхождения"), а также рисунок конечного автомата по заданным данным.

1.1.Уточнения требований после сдачи 1-ой версии

Добавить визуализацию

Добавить удаление вершин

1.2.Уточнения требований после сдачи 2-ой версии

2. ПЛАН РАЗРАБОТКИ И РАСПРЕДЕЛЕНИЕ РОЛЕЙ В БРИГАДЕ

2.1. План разработки

6 июля: согласование спецификации и плана разработки.

7 июля: создание прототипа программы(приложения, демонстрирующего интерфейс, но не реализующее основные функции);

12 июля: создание первой версии программы с возможностью добавлять переход по состоянию

13 июля: создание второй версии программы с добавленной возможностью менять терминальное состояние и возможностью удалять сами состояния

13 июля: сдача финальной версии с отчётом

2.2. Распределение ролей в бригаде

Голов – общение с преподавателем, оформление документов, алгоритмическая часть

Тоун – Работа с графическим интерфейсом, оформление документов.

3. РЕАЛИЗАЦИЯ

3.1. Структуры данных

Класс *Bor()* – представляет собой класс для создания и модификаций бора, реализован в виде дерева.

Имеет следующие поля:

root: Node? – корень бора

totalString: Int - количество терминальных состояний

sufflink: Node? – суффиксная ссылка

terminal: Boolean = false – флаг терминальной ссылки

go: MutableMap – список возможных переходов по символу

children: MutableList<Pair?> - массив детей, представленных в виде пары значений – узел и символ перехода к нему

subPatterns : MutableList - все Node в которые можно прийти передвигаясь по суффиксным ссылкам от текущей ноды до корня

flagSubPatterns: Boolean – флаг проверки, заполнен ли массив subPatterns полностью.

Класс Node() – представляет собой вершину бора.

Имеет следующие поля:

Number: Int? – номер терминального состояния.

Father: Node? – отец

Pchar: Char? символ вершины

sufflink: Node?

terminal : Bool

go: MutableMap<Char, Node>

children: MutableList<Pair<Node, Char>>?

subPatterns = mutableSetOf<Node>()

flagSubPatterns = false

Класс Send() – набор функций и полей для работы пользовательского интерфейса

Имеет следующие поля:

txt: TextField – ввод текста
patterns: TextField- ввод шаблонов
nodeInput: TextField- ввод вершины на добавление
terminalInput: TextField- ввод вершины на смену терминальности
textField: TextField- смена текста
wrong: Text- сообщение об ошибке
textText: Text- индикатор текста
patternsText: Text- шаблоны
answerText: Text- ответ
sendButton: Button- кнопка подтверждения текста и шаблонов
addButton: Button- кнопка добавления вершины
switchButton: Button- кнопка переключения терминальности
backButton: Button- кнопка возвращения к начальному окну
switchTextButton: Button- кнопка переключения текста
trieImg: ImageView- изображения бора
fsmImg: ImageView- изображения конечного автомата
bor: Bor- бор для работы алгоритма

Класс AutomatonNode: - информация о вершинах изображения автомата

children: MutableMap<Char, AutomatonNode> -- потомки
suffixLink: AutomatonNode? – суффиксные ссылки
outputLink: AutomatonNode? – терминальные ссылки
isTrue: Boolean – терминальность

3.2. Основные методы

Методы класса Node():

getNode(ch:Char) – возвращает потомка по символу ch
addChildren(ch:Char) – добавляет ребёнка по символу ch
getPerentChar()- возвращает символ вершины
getWord() – возвращает всё слово
getWordLen()- возвращает высоту вершины

toString() – выводит вершину

getChildren() – возвращает потомков

allSubPatterns() – возвращает множество узлов, собранных из массивов subPatterns узлов встретившихся на пути обхода.

Методы класса Bor():

insert(string: String) – добавляет в бор вершину по шаблону

go(node: Node, char: Char) – возвращает вершину по которой можно перейти по символу char

sufflinkOf(node: Node) – находит суффиксную ссылку узла

getIndexesOf(text: String) – выводит ответ

toString() – вывод бора в текстовом виде

Методы класса Send():

sendData() – переводит приложение в состояние ответа если данные введены корректно, иначе выводит ошибку

activateAlg() – выводит ответ на экран

rebuildBor() – перестройка бора по шаблонам

switchText() – смена текста

addNode() – добавление вершины в бор

deletePattern(str: String) – удаления шаблона из набора шаблонов

goodSwitchTerminal() – смена терминального состояния

activatePng() – обновляет картинку на экране

buildAutomaton(patterns: List<String>) – строит бор на основе списка шаблонов

markTrueNodes(root: AutomatonNode) – помечает терминальные вершины

saveTreAsPng(root: AutomatonNode, filename: String, path: String) – сохраняет дерево как картинку

buildAutomat(patterns: List<String>) – строит автомат на основе шаблонов

saveAutomatonAsPng(automaton: AutomatonNode, filename: String,

path: String) – сохраняет автомат как картинку

back() – смена окна на начальное

4. Тестирования

Тест на отсутствие введенных данных	
Входные данные:	Текст: Шаблоны:
Ожидаемый результат:	Алгоритм должен указать на ошибку ввода
Полученный результат:	<div><p>Aho-Corasick</p><p>there is no text</p><div><div>Впишите ваш текст</div><div>Впишите ключевые слова</div><div>Отправить</div></div><p>Корректное выполнение работы.</p></div>
Тест на один шаблон	
Входные данные:	Текст: abacaba Шаблоны: a

Ожидаемый результат:	Алгоритм должен вернуть серию вхождения шаблона в текст и вывести бор с автоматом
Полученный результат:	<div><h1>Aho-Corasick</h1><div><div>Text: <input type="text" value="abacaba"/></div><div>поменять</div><div>Patterns: a</div><div><div><div><div></div><div>b</div><div>ba</div><div>a</div></div><div><div>a</div><div>a</div><div>a</div><div>a</div></div><div><div>a</div><div>a</div><div>a</div><div>a</div></div></div><div><div></div><div>a</div></div></div><div><div>add new node</div><div>switch terminal of</div></div><div><div>Button</div><div>Button</div></div><div>Answer 1 1 1 3 1 5 1 7</div><p>Корректное выполнение работы.</p></div></div>
Тест на граф с большим количеством вершин и ребер	
Входные данные:	Текст abacaba Шаблоны a#ab#aba#ac#acb
Ожидаемый результат:	Алгоритм должен вернуть серию вхождения шаблона в текст и вывести бор с автоматом
Полученный результат:	

Aho-Corasick

Text:

Patterns: a#ab#aba#ac#acb

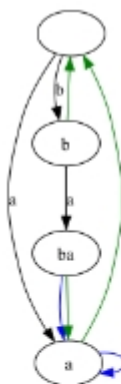
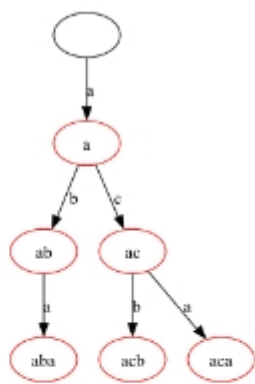
Answer

1	1
1	3
1	5
1	7
2	1
2	5
3	1
3	5
4	5

Корректное выполнение работы.

Тест на добавление вершины

Входные данные:	Ввод в окно с подсказкой «add new node» ac->a
Ожидаемый результат:	Алгоритм должен расширить бор и изменить ответ

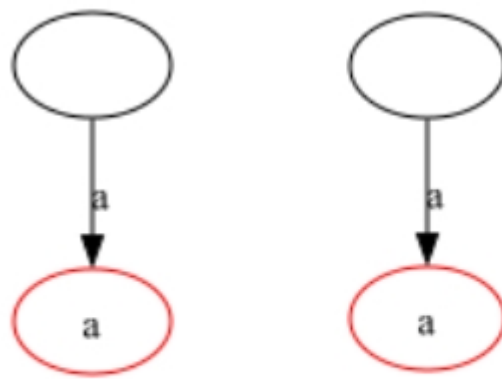
Полученный результат:	<div style="text-align: center;"> <h1>Aho-Corasick</h1> </div> <div style="display: flex; justify-content: space-between; align-items: flex-start; padding: 10px;"> <div style="width: 45%;"> <p>Text: <input type="text" value="другой текст"/></p> <p><input type="button" value="поменять"/></p> <p>Patterns: a#ab#abc#aba</p>  <p>ab->a</p> <p><input type="button" value="Button"/></p> </div> <div style="width: 45%;"> <p>Answer</p> <p>1 1</p> <p>1 3</p> <p>1 5</p> <p>1 7</p> <p>2 1</p> <p>2 5</p> <p>4 1</p> <p>4 5</p>  <p>switch terminal of</p> <p><input type="button" value="Button"/></p> <p><input type="button" value="back"/></p> </div> </div> <p style="text-align: center; margin-top: 20px;">Корректное выполнение работы.</p>
Тест на смену терминального состояния	
Входные данные:	<p>Текст abacaba</p> <p>Шаблоны a#ab#aba#ac#acb</p> <p>смена состояния: aba</p>
Ожидаемый результат:	Алгоритм должен изменить ответ и поменять бор
Полученный результат:	

Aho-Corasick

Text: Answer
1 1
1 3

ПОМЕНЯТЬ

Patterns: a



add new node

ba

Button

Button

back

Корректное выполнение работы.

4.1. Тестирование граничных условий

Тест на пустой текст и шаблон	
Входные данные:	Текст Шаблоны
Ожидаемый результат:	Алгоритм должен указать на ошибку

Полученный результат:	<div><div>Aho-Corasick</div><div>there is no text</div><div><div>Впишите ваш текст</div><div>Впишите ключевые слова</div><div>Отправить</div></div><div>Корректное выполнение работы.</div></div>
Тест на пустой шаблон	
Входные данные:	Текст: фтфа Шаблоны:
Ожидаемый результат:	Алгоритм должен указать на ошибку

Полученный результат:	<p><i>Aho-Corasick</i></p> <p>there is no patterns</p> <p>пвапва</p> <p>Впишите ключевые слова</p> <p>Отправить</p> <p>Корректное выполнение работы.</p>
-----------------------	---

4.2 Тестирование интерфейса

Тест на добавление вершины в граф	
Входные данные:	Шаблоны ab#ba Добавление: ba->a
Ожидаемый результат:	Алгоритм должен Добавить в бор и автомат новую вершину

Полученны
й
результат:

Answer

1 1
1 5
1 5
2 2
2 4

Text:

Patterns: ab#ba#baa

Корректное выполнение работы.

Тест на удаление вершины из графа

Входные
данные:

Шаблоны: ab#ba#baa

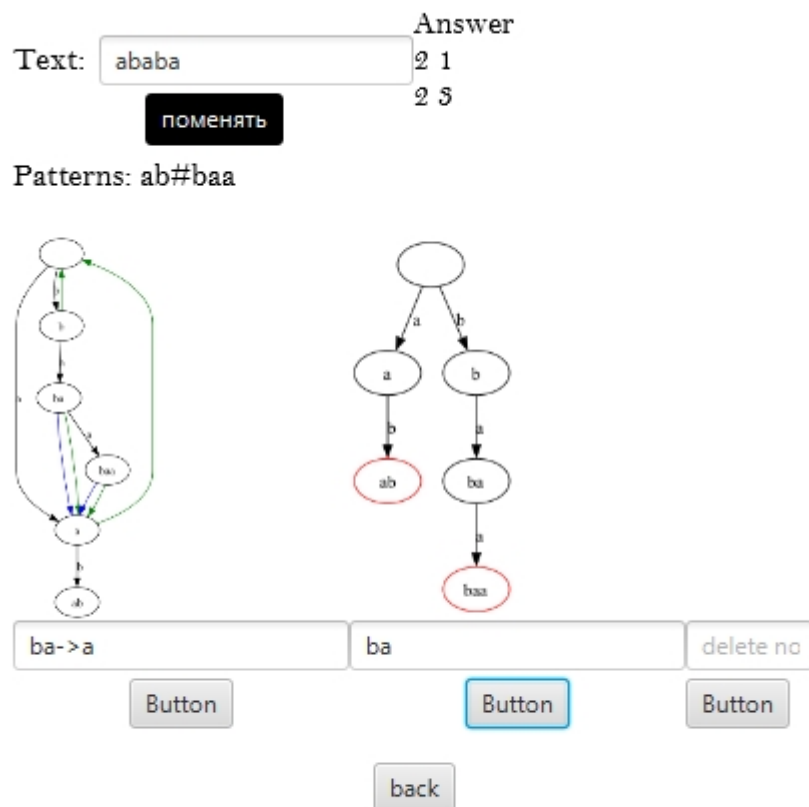
Удалённая вершина: baa

Ожидаемы
й
результат:

Алгоритм должен удалить вершину

Полученны й результат:	<div><div><div>Answer</div><div>1 1</div><div>1 3</div><div>1 5</div><div>2 2</div><div>2 4</div></div><div>Text: <input type="text" value="ababab"/></div><div>ПОМЕНЯТЬ</div><div>Patterns: ab#ba</div><div><div><div><div><div></div><div>b</div><div>ba</div><div>a</div><div>ab</div></div><div><div>a</div><div>b</div><div>ba</div><div>a</div><div>ab</div></div><div><div>a</div><div>b</div><div>ba</div><div>a</div><div>ab</div></div></div><div><div>ba->a</div><div>switch terminal of</div><div>baa</div></div><div><div>Button</div><div>Button</div><div>Button</div></div></div><div>Корректное выполнение работы.</div></div></div>
Тест на смену терминального состояния	
Входные данные:	Шаблоны: ab#ba#baa Состояние: ba
Ожидаемы й результат:	Алгоритм должен сменить терминальность

Полученны
й
результат:



Корректное выполнение работы.

-

- **Ручное тестирование**

- Проверка навигации и функциональности: проход по всем разделам и функциям интерфейса, чтобы убедиться, что они работают правильно.
- Проверка отображения элементов: убедиться, что все элементы интерфейса правильно отображаются (тексты, изображения, кнопки и т. д.).
- Проверка взаимодействия: протестировать взаимодействие с интерфейсом (нажимать кнопки, заполнять формы, выполнять действия) и убедиться, что все работает ожидаемым образом.

4.3 Тестирование структуры данных

- Тест на добавление вершины в граф:
 - Входные данные: пустой граф, команда добавления вершины.
 - Ожидаемый результат: вершина успешно добавлена в граф.
- Тест на удаление вершины из графа:
 - Входные данные: граф с несколькими вершинами, команда удаления одной из вершин.
 - Ожидаемый результат: вершина успешно удалена из графа и связанные с ней ребра также удалены.
- Тест автомата на набор с большим количеством похожих, но не равных шаблонов:

Входные данные: граф с несколькими вершинами, команда удаления одной из вершин.

 - Ожидаемый результат: вывод графа и бора с корректными терминальными суффиксами и ссылками.
- Тест на удаление терминальных ссылок при смене терминального состояния:
 - Входные данные: вершина , команда смена терминального состояния.
 - Ожидаемый результат: конечный автомат перестроен, в боре цвет поменялся, добавился\удалился шаблон.
- Тест на бор и конечный автомат из одного шаблона:
 - Входные данные: текст и шаблоны
 - Ожидаемый результат: конечный автомат и бор представляют собой однонаправленный список с единственным терминальным состоянием в конце.
- Тест на бор и конечный автомат из шаблонов, которые являются префиксом другого шаблона:
 - Входные данные: текст и шаблоны, каждый шаблон является подмножеством другого шаблона.

- Ожидаемый результат: конечный автомат и бор представляют собой однонаправленный список с единственным терминальным состоянием в разных местах.

ЗАКЛЮЧЕНИЕ

В результате выполнения данной работы было разработано приложение на языке Kotlin, которое реализует алгоритм Ахо-Корасик с использованием, созданного с помощью библиотеки JavaFX графического интерфейса. Приложение создаёт визуальное представление бора и конечного автомата.

В ходе выполнения работы были получены навыки работы с языком программирования Kotlin, компиляции исполняемых файлов, а также получены навыки профессиональной разработки продуктов.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Алгоритм Ахо-Корасик Вики конспекты. URL:
https://neerc.ifmo.ru/wiki/index.php?title=Алгоритм_Ахо-Корасик
2. Kotlin docs. URL:
<https://kotlinlang.org/docs/home.html>.
3. JavaFX DOCUMENTATION. URL:
<https://openjfx.io/>.
4. Курс введение в Kotlin JVM. URL:
<https://stepik.org/course/5448/syllabus> .

ПРИЛОЖЕНИЕ А

КОД ПРОГРАММЫ

Файл Main.kt

```
package com.example.aho

import javafx.application.Application
import javafx.fxml.FXMLLoader
import javafx.scene.Scene
import javafx.stage.Stage

class HelloApplication : Application() {

    private var stage: Stage? = null //oho

    override fun start(stage1: Stage?) {
        this.stage = stage1
        val fxmlLoader =
FXMLLoader(HelloApplication::class.java.getResource("he
llo-view.fxml"))
        val scene = Scene(fxmlLoader.load(), 800.0,
800.0)
        stage1?.title = "aho-corasick"
        stage1?.scene = scene
        stage1?.show()

    }
}

fun main() {
    Application.launch(HelloApplication::class.java)
}
```

Файл Send.kt

```
package com.example.aho

import guru.nidi.graphviz.attribute.Color
import guru.nidi.graphviz.attribute.Label
import guru.nidi.graphviz.engine.Format
import guru.nidi.graphviz.engine.Graphviz
import guru.nidi.graphviz.model.Factory
import guru.nidi.graphviz.model.MutableNode
import javafx.fxml.FXML
import javafx.scene.control.Button
```

```

import javafx.scene.control.TextField
import javafx.scene.image.ImageView
import javafx.scene.text.Text
import java.io.File

class Bor {
    /**
     * Класс вершины бора
     */
    class Node(
        var number: Int? = null,
        val father: Node? = null,
        protected val pchar: Char? = null
    ) {
        var sufflink: Node? = null
        var terminal = false
        var go: MutableMap<Char, Node> = mutableMapOf()
        private var children: MutableList<Pair<Node,
Char>>? = null
        var subPatterns = mutableSetOf<Node>()
        var flagSubPatterns = false

        /**
         * Возвращает true если существует дочерний
Node c
символом-аргументов ведущим к нему
         */
        operator fun contains(sym: Char): Boolean {
            if (children == null) return false
            return children!!.any {
                it.second == sym
            }
        }

        /**
         * Возвращает объект Node из массива детей,
если к нему
ведет символ поиска (аргумент)
         * Возвращает null при отсутствии детей с
запрашиваемым
символом
         */
        fun getNode(ch: Char): Node? {
            if (children == null) { // нет потомков
                return null
            }

```

```

        } else { // найти потомка к которому можно
прийти по символу
            return children!!.firstOrNull {
                it.second == ch
            }?.first
        }
    }

    /**
    * Возвращает объект Node в который переходят
по
символу-аргументу
    */

    fun addChildren(ch: Char): Node {
        // еще нет ребенка с таким символом
        if (ch !in this) {
            // проверка на наличие дочернего
массива
            if (children == null)
                children = mutableListOf()
            // создание новой Node и добавление его
в массив
            val newNode = Node(father = this, pchar
= ch)
            children!!.add(Pair(newNode, ch))
            return newNode
        } else {
            // вернуть существующий Node
            return getNode(ch)!!
        }
    }

    /**
    * Возвращает символ от отца
    */
    fun getPerentChar(): Char? = pchar

    /**
    * -Получение текста для состояния
    */
    fun getWord(): String {
        val string: String
        if (father == null)

```



```

        string = ""
    else
        string = father.getWord() + pchar
    return string
}

fun getWordLen(): Pair<Int, Int?> {
    val len: Int
    if (father == null) {
        len = 0
    } else {
        len = father.getWordLen().first + 1
    }
    return Pair(len, number)
}

override fun toString(): String {
    var result = "father is "
    result += "${father?.getWord() ?: "Null father"} ->${pchar}\n"
    result += "Terminal: ${terminal}\n"
    //result += "Index in
sample:${indexes?.joinToString(separator = " ")}\n"
    result += "Sub
patterns:${subPatterns.joinToString(separator = "\t")} {
it.getWord() }\n"
    result += "children of
${this.getWord()}\n"
    if (children == null) result += "null"
    else children!!.forEach {
        result += "${it.second}-
${it.first.getWord()}\t"
    }
    return result + "\n"
}

fun getChildren(): MutableList<Pair<Node,
Char>>? = children

fun allSubPatterns(): MutableSet<Node> {
    if (!flagSubPatterns) { // в массив
subPatterns еще не были добавлены все внутренние
паттерны
        sufflink?.let {

```

```

subPatterns.addAll(it.allSubPatterns())
    }
    flagSubPatterns = true
}
return subPatterns
}
}

var root = Node()
var totalString: Int = 0

/**
 * Вставляет строку в бор
 * устанавливает значение для терминальной вершины
 */
fun insert(string: String) {
    totalString++
    var currentNode = root
    string.forEachIndexed { index, c ->
        if (c !in currentNode) {
            currentNode.addChildren(c)
        }
        currentNode = currentNode.getNode(c)!!
    }
    currentNode.number = totalString
    currentNode.terminal = true
    currentNode.subPatterns.add(currentNode)
}

/**
 * Используется для построения суффиксных ссылок
 */
fun go(node: Node, char: Char): Node {
    if (!node.go.contains(char)) {
        if (node.getNode(char) != null) {
            node.go[char] = node.getNode(char)!!
        } else if (node == root) { // перехода по
ребру бора нет, находимся в корне
            node.go[char] = root
        } else { //нужно перейти по суффиксной
ссылке и оттуда искать
            node.go[char] = go(sufflinkOf(node),
char)
        }
    }
}

```

```

        }
        return node.go[char]!!
    }

    /**
     * Находит суффиксную ссылку узла
     */
    private fun sufflinkOf(node: Node): Node {
        if (node.sufflink == null) {
            if (node == root || node.father == root) {
                node.sufflink = root
            } else { // переход по символу
                node.sufflink =
go(sufflinkOf(node.father!!), node.getPerentChar()!!)
            }
        }
        return node.sufflink!!
    }

    /**
     * Возвращает массив пар:Номер слова, Индекс в
тексте
     */
    fun getIndexesOf(text: String): List<Pair<Int,
Int>> {
        val result = mutableSetOf<Pair<Int,
Int>>() //формочка для вывода
        var node = root
        for (i in text.indices) {
            node = go(node, text[i]) //переход по
суффиксной ссылке
            var tmpNode = node
            while (tmpNode != root) {
                tmpNode = sufflinkOf(tmpNode)
            }

            node.allSubPatterns().forEach {
                val value = it.getWordLen()
                result.add(
                    Pair(i - value.first + 2,
value.second!!)
                )
            }
        }
    }

```

```

        return result.sortedWith(compareBy({ it.second
    }, { it.first })))
    }

```

```

    fun toString(node: Node = root, tab: Int = 0):
String {
    var result = node.toString()
    node.getChildren()?.forEach {
        result += "\n"
        result += toString(it.first, tab +
1).prependIndent("\t".repeat(tab))
    }
    return result
}
}

```

```

class Send {

    @FXML
    private lateinit var txt: TextField

    @FXML
    private lateinit var patterns: TextField

    @FXML
    private lateinit var nodeInput: TextField

    @FXML
    private lateinit var terminalInput: TextField

    @FXML
    private lateinit var textField: TextField

    @FXML
    private lateinit var delInput: TextField

    @FXML
    private lateinit var wrong: Text

    @FXML
    private lateinit var textText: Text

    @FXML
    private lateinit var patternsText: Text

```

```

@FXML
private lateinit var answerText: Text

@FXML
private lateinit var sendButton: Button

@FXML
private lateinit var addButton: Button

@FXML
private lateinit var switchButton: Button

@FXML
private lateinit var backButton: Button

@FXML
private lateinit var switchTextButton: Button

@FXML
private lateinit var delButton: Button

@FXML
private lateinit var trieImg: ImageView

@FXML
private lateinit var fsmImg: ImageView

private lateinit var bor: Bor
//private lateinit var graph

@FXML
private fun sendData() {

    //wrong.text ="oops"
    if (txt.text.isNotEmpty() &&
patterns.text.isNotEmpty()) {
        //
        //println(txt.text)
        textField.isVisible = true
        switchTextButton.isVisible = true
        txt.isVisible = false
        patterns.isVisible = false
        patterns.isVisible = false
        sendButton.isVisible = false
    }
}

```

```

        nodeInput.isVisible = true
        terminalInput.isVisible = true
        addButton.isVisible = true
        switchButton.isVisible = true
        trieImg.isVisible = true
        fsmImg.isVisible = true
        backButton.isVisible = true
        textText.isVisible = true
        patternsText.isVisible = true
        answerText.isVisible = true
        delInput.isVisible = true
        delButton.isVisible = true
        activateAlg()
    } else if (txt.getText().isEmpty()) wrong.text
= "there is no text"
        else if (patterns.getText().isEmpty())
wrong.text = "there is no patterns"
    }

    class AutomatonNode(val state: String, val
transition: Char) {
        val children: MutableMap<Char, AutomatonNode> =
mutableMapOf()
        var suffixLink: AutomatonNode? = null
        var outputLink: AutomatonNode? = null
        var isTrue: Boolean = false
    }

    private fun buildAutomaton(patterns: List<String>):
AutomatonNode {
        val root = AutomatonNode(" ", ' ')

        for (pattern in patterns) {
            var currentNode = root

            for (char in pattern) {
                if
(!currentNode.children.containsKey(char)) {
                    val newNode =
AutomatonNode(currentNode.state + char, char)
                    currentNode.children[char] =
newNode
                }
            }
        }
    }

```

```

        currentNode =
currentNode.children[char]!!
    }

    currentNode.outputLink = currentNode
    currentNode.isTrue = true
}

val queue = mutableListOf<AutomatonNode>()

for (child in root.children.values) {
    child.suffixLink = root
    queue.add(child)
}

while (queue.isNotEmpty()) {
    val currentNode = queue.removeAt(0)

    for (child in currentNode.children.values)
{
        queue.add(child)

        var suffixNode = currentNode.suffixLink

        while (suffixNode != null &&
!suffixNode.children.containsKey(child.transition)) {
            suffixNode = suffixNode.suffixLink
        }

        child.suffixLink =
suffixNode?.children?.get(child.transition) ?: root
        child.outputLink =
child.suffixLink?.outputLink ?: child.suffixLink
    }
}

return root
}

private fun markTrueNodes(root: AutomatonNode) {
    val visited = mutableSetOf<AutomatonNode>()

    fun dfs(node: AutomatonNode) {
        if (node.isTrue) {
            visited.add(node)

```

```

        return
    }

    if (visited.contains(node)) {
        return
    }

    visited.add(node)

    for (child in node.children.values) {
        dfs(child)
    }

    val suffixNode = node.suffixLink
    if (suffixNode != null) {
        dfs(suffixNode)
    }
}

dfs(root)
}

private fun saveTreAsPNG(root: AutomatonNode,
filename: String, path: String = "") {
    val graph =
Factory.mutGraph().setDirected(true)
    val nodes = mutableMapOf<AutomatonNode,
MutableNode>()

    fun buildGraph(node: AutomatonNode) {
        val mutableNode = nodes.getOrPut(node) {
Factory.mutNode(node.state) }
        if (node.isTrue) {
            mutableNode.add(Color.RED)
        }
        graph.add(mutableNode)

        for (child in node.children.values) {
            buildGraph(child)

            val childNode = nodes[child]!!
            val link =
Factory.to(childNode).with(Label.of(child.transition.to
String()))
            mutableNode.addLink(link)

```



```

        }
    }

    buildGraph(root)

    val dot =
Graphviz.fromGraph(graph).render(Format.DOT).toString()
        val file = File(path, filename)
        val renderedGraph =
Graphviz.fromString(dot).render(Format.PNG).toFile(file
    )
        println("Бор сохранен в файл:
${file.absolutePath}")
    }

    private fun buildAutomat(patterns: List<String>):
AutomatonNode {
        val root = AutomatonNode(" ", ' ')

        for (pattern in patterns) {
            var currentNode = root

            for (char in pattern) {
                if
(!currentNode.children.containsKey(char)) {
                    val newNode =
AutomatonNode(currentNode.state + char, char)
                    currentNode.children[char] =
newNode
                }

                currentNode =
currentNode.children[char]!!
            }

            currentNode.outputLink = currentNode
        }

        val queue = mutableListOf<AutomatonNode>()

        for (child in root.children.values) {
            child.suffixLink = root
            queue.add(child)
        }
    }

```

```

        while (queue.isNotEmpty()) {
            val currentNode = queue.removeAt(0)

            for (child in currentNode.children.values)
            {
                queue.add(child)

                var suffixNode = currentNode.suffixLink

                while (suffixNode != null &&
!suffixNode.children.containsKey(child.transition)) {
                    suffixNode = suffixNode.suffixLink
                }

                child.suffixLink =
suffixNode?.children?.get(child.transition) ?: root
                child.outputLink =
child.suffixLink?.outputLink ?: child.suffixLink
            }
        }

        return root
    }

    private fun saveAutomatonAsPNG(automaton:
AutomatonNode, filename: String, path: String =
"src/main/resources/com/example/aho") {
        val graph =
Factory.mutableGraph().setDirected(true)
        val nodes = mutableMapOf<AutomatonNode,
MutableNode>()

        fun buildGraph(node: AutomatonNode) {
            val mutableNode = nodes.getOrPut(node) {
Factory.mutableNode(node.state) }
            graph.add(mutableNode)

            for (child in node.children.values) {
                buildGraph(child)

                val childNode = nodes[child]!!
                val link =
Factory.to(childNode).with(Label.of(child.transition.to
String()))
                mutableNode.addLink(link)
            }
        }
    }

```

```

    }

    node.outputLink?.let { outputLink ->
        nodes[outputLink]?.let { outputLinkNode
->
            val outputLinkLink =
Factory.to(outputLinkNode).with(Label.of("")).add(Color
.BLUE)
            mutableNode.addLink(outputLinkLink)
        }
    }

    node.suffixLink?.let { suffixLink ->
        nodes[suffixLink]?.let { suffixLinkNode
->
            val suffixLinkLink =
Factory.to(suffixLinkNode).with(Label.of("")).add(Color
.GREEN)
            mutableNode.addLink(suffixLinkLink)
        }
    }
}

buildGraph(automaton)

val dot =
Graphviz.fromGraph(graph).render(Format.DOT).toString()
val file = File(path, filename)
val renderedGraph =
Graphviz.fromString(dot).render(Format.PNG).toFile(file
)
    println("Автомат сохранен в файл:
${file.absolutePath}")
}

@FXML
private fun activatePng(){
    textText.text = "Text: "
    patternsText.text = "Patterns:
${patterns.text}"
    val patterns = "${patterns.text}"
    val automaton1 =
buildAutomat(patterns.split("#"))
    saveAutomatonAsPNG(automaton1, "fsm.png")
    val patterns1 = patterns.split("#")

```

```

        val automaton2 = buildAutomaton(patterns1)
        markTrueNodes(automaton2)
        val filename1 = "trie.png"
        val path1 =
"src/main/resources/com/example/aho"
        val path = if (path1.isNotBlank()) path1 else
"src/main/resources/com/example/aho" // Если путь не
указан, сохранить файл в текущей директории

        saveTreAsPNG(automaton2, filename1, path1)

        val relativePath2 =
"src/main/resources/com/example/aho" // Относительный
путь от текущей директории

        val path2 =
File(System.getProperty("user.dir"),
relativePath2).absolutePath
        val newImage = ImageView("$path2/fsm.png")
        val newImage2 = ImageView("$path2/trie.png")
        trieImg.image = newImage.image
        fsmImg.image = newImage2.image

    }

    @FXML
    private fun activateAlg() {

        textText.text = "Text: "
        textField.text = txt.text
        patternsText.text = "Patterns:
${patterns.text}"
        rebuildBor()
        answerText.text = "Answer
\n${bor.getIndexesOf(txt.text).joinToString("\n")} {
"${it.second} ${it.first}" }"
        activatePng()
    }
    //
    private fun rebuildBor() {
        bor = Bor()
        for (word in patterns.text.split('#')) {
            bor.insert(word)
        }
    }
}

```

```

        private fun deletePattern(str: String) {
            var tmpText =
                "${patterns.text}#".replace("#${str}#",
                "#").replace("##", "#")
            println(tmpText)
            if (tmpText.first() == '#') {
                tmpText = tmpText.substring(1)
            }
            if (tmpText.isNotEmpty() && tmpText.last() ==
                '#') {
                tmpText = tmpText.substring(0,
                tmpText.length - 1)
            }
            patterns.text = tmpText
            patternsText.text = "Patterns:
            ${patterns.text}"
        }

        @FXML
        private fun switchText() {
            answerText.text =
                "Answer
            \n${bor.getIndexesOf(textField.getText()).joinToString(
                "\n") { "${it.second} ${it.first}" }}"

        }

        @FXML
        private fun delNode() {
            val oldNode = delInput.text
            var pts =
                patterns.text.split("#").toMutableList()
            var i = 0

            while (i != pts.size) {
                if (oldNode in pts[i]) {
                    println(i)
                    println(pts)
                    deletePattern(pts[i])
                    pts.removeAt(i)
                    i--
                }
                i++
            }
        }

```

```

        rebuildBor()
        answerText.text =
            "Answer
\n${bor.getIndexesOf(txt.text).joinToString("\n")} {
"${it.second} ${it.first}" }"
        activatePng()
    }

```

```

@FXML
private fun addNode() {
    val newNodeText =
nodeInput.getText().toString().split("->")
    val fullNewNode: String = newNodeText[0] +
newNodeText[1]
    if (fullNewNode !in patternsText.text) {
        patternsText.text += "#$fullNewNode"
        rebuildBor()
        //вывод картинки
        patterns.text += "#$fullNewNode"
        bor.insert(fullNewNode)
        //вывод картинки
        answerText.text =
            "Answer
\n${bor.getIndexesOf(txt.text).joinToString("\n")} {
"${it.second} ${it.first}" }"
    }
    activatePng()
}

```

```

@FXML
private fun goodSwitchTerminal() {
    //rebuildBor()
    var tmpNode = bor.root
    for (i in terminalInput.text) tmpNode =
bor.go(tmpNode, i)
    tmpNode.terminal = !tmpNode.terminal
    if (tmpNode.terminal) {
        tmpNode.subPatterns.add(tmpNode)
        tmpNode.number = ++bor.totalString

        //tmpNode.number=patterns.text.count{it ==
'#' }+2
        //bor.totalString++
    }
}

```

```

        patterns.text += "#${terminalInput.text}"
        patternsText.text = "Patterns:
${patterns.text}"
    } else {
        tmpNode.subPatterns.remove(tmpNode)
        tmpNode.number = null
        --bor.totalString
        deletePattern(terminalInput.text)
    }
    answerText.text =
        "Answer
\n${bor.getIndexesOf(txt.text).joinToString("\n") {
    "${it.second} ${it.first}" }}"
        activatePng()
    }

@FXML
private fun back() {
    txt.text = ""
    txt.isVisible = true
    patterns.text = ""
    patterns.isVisible = true
    sendButton.isVisible = true
    nodeInput.isVisible = false
    terminalInput.isVisible = false
    addButton.isVisible = false
    switchButton.isVisible = false
    trieImg.isVisible = false
    fsmImg.isVisible = false
    backButton.isVisible = false
    textField.isVisible = false
    switchTextButton.isVisible = false
    textText.isVisible = false
    patternsText.isVisible = false
    answerText.isVisible = false

    delInput.isVisible = false
    delButton.isVisible = false
}
}

```

Файл hello-view.fxml

```

<?xml version="1.0" encoding="UTF-8"?>

<?import javafx.geometry.Insets?>
<?import javafx.scene.Cursor?>
<?import javafx.scene.control.Button?>
<?import javafx.scene.control.Label?>
<?import javafx.scene.control.TextField?>
<?import javafx.scene.image.Image?>
<?import javafx.scene.image.ImageView?>
<?import javafx.scene.layout.ColumnConstraints?>
<?import javafx.scene.layout.GridPane?>
<?import javafx.scene.layout.RowConstraints?>
<?import javafx.scene.layout.VBox?>
<?import javafx.scene.text.Font?>
<?import javafx.scene.text.Text?>

<VBox alignment="TOP_CENTER" prefWidth="951.0"
spacing="20.0" style="-fx-background-color: #FFFFFF;"
xmlns="http://javafx.com/javafx/20.0.1"
xmlns:fx="http://javafx.com/fxml/1"
fx:controller="com.example.aho.Send">
    <padding>
        <Insets bottom="50.0" left="200.0"
right="200.0" top="50.0" />
    </padding>
    <Text strokeType="INSIDE" strokeWidth="0.0"
text="Aho-Corasick" textAlignment="CENTER"
wrappingWidth="248.0">
        <font>
            <Font name="Segoe Print" size="32.0" />
        </font>
    </Text>
    <Text id="wrong" fx:id="wrong" strokeType="OUTSIDE"
strokeWidth="0.0">
        <font>
            <Font name="System Bold" size="21.0" />
        </font>
    </Text>

    <Label fx:id="welcomeText" />
    <GridPane prefHeight="131.0" prefWidth="551.0">
        <columnConstraints>
            <ColumnConstraints hgrow="SOMETIMES"
minWidth="10.0" prefWidth="100.0" />

```



```

        <ColumnConstraints hgrow="SOMETIMES"
minWidth="10.0" prefWidth="100.0" />
    </columnConstraints>
    <rowConstraints>
        <RowConstraints minHeight="10.0"
prefHeight="30.0" vgrow="SOMETIMES" />
        <RowConstraints minHeight="10.0"
prefHeight="30.0" vgrow="SOMETIMES" />
        <RowConstraints minHeight="10.0"
prefHeight="30.0" vgrow="SOMETIMES" />
    </rowConstraints>
    <children>
        <Text fx:id="patternsText"
strokeType="OUTSIDE" strokeWidth="0.0" text="Patterns:
" visible="false" GridPane.rowIndex="2">
            <font>
                <Font name="Bell MT" size="16.0" />
            </font>
        </Text>
        <Text fx:id="answerText"
strokeType="OUTSIDE" strokeWidth="0.0" text="Answer: "
visible="false" GridPane.columnIndex="1">
            <font>
                <Font name="Bell MT" size="16.0" />
            </font>
        </Text>
        <GridPane>
            <columnConstraints>
                <ColumnConstraints
hgrow="SOMETIMES" minWidth="10.0" prefWidth="100.0" />
                <ColumnConstraints
hgrow="SOMETIMES" minWidth="10.0" />
            </columnConstraints>
            <rowConstraints>
                <RowConstraints minHeight="10.0"
vgrow="SOMETIMES" />
            </rowConstraints>
            <children>
                <TextField fx:id="textField"
prefHeight="25.0" prefWidth="214.0" promptText="другой
текст" visible="false" GridPane.columnIndex="1" />
                <Text fx:id="textText"
strokeType="OUTSIDE" strokeWidth="0.0" text="Text: "
visible="false">
                    <font>

```



```

        <Image url="@trie.png" />
    </image></ImageView>
    <TextField fx:id="nodeInput"
promptText="add new node" visible="false"
GridPane.rowIndex="1" />
    <TextField fx:id="terminalInput"
prefHeight="25.0" prefWidth="133.0" promptText="switch
terminal of" visible="false" GridPane.columnIndex="1"
GridPane.rowIndex="1" />
    <Button fx:id="addButton"
alignment="CENTER" contentDisplay="CENTER"
mnemonicParsing="false" onAction="#addNode" style="-fx-
alignment: center;" text="Button"
textAlignment="CENTER" visible="false"
GridPane.halignment="CENTER" GridPane.rowIndex="2"
GridPane.valignment="CENTER">
        <cursor>
            <Cursor fx:constant="OPEN_HAND" />
        </cursor>
        <GridPane.margin>
            <Insets />
        </GridPane.margin>
    </Button>
    <Button fx:id="switchButton"
mnemonicParsing="false" onAction="#goodSwitchTerminal"
text="Button" visible="false" GridPane.columnIndex="1"
GridPane.halignment="CENTER" GridPane.rowIndex="2"
GridPane.valignment="CENTER">
        <GridPane.margin>
            <Insets />
        </GridPane.margin>
    </Button>
    <TextField fx:id="delInput"
prefHeight="25.0" prefWidth="133.0" promptText="delete
node" visible="false" GridPane.columnIndex="2"
GridPane.rowIndex="1" />
    <Button fx:id="delButton"
mnemonicParsing="false" onAction="#delNode"
text="Button" visible="false" GridPane.columnIndex="2"
GridPane.rowIndex="2" />
</children>
</GridPane>
    <Button fx:id="backButton" mnemonicParsing="false"
onAction="#back" text="back" visible="false" />

```

```
<TextField id="txt" fx:id="txt" prefHeight="40.0"
prefWidth="368.0" promptText="Впишите ваш текст"
style="-fx-background-color: #000000;">
    <font>
        <Font size="14.0" />
    </font></TextField>
    <TextField id="patterns" fx:id="patterns"
prefHeight="40.0" prefWidth="368.0" promptText="Впишите
ключевые слова" style="-fx-background-color: #000000;">
    <font>
        <Font size="14.0" />
    </font></TextField>
    <Button id="send" fx:id="sendButton"
mnemonicParsing="false" onAction="#sendData"
prefHeight="42.0" prefWidth="99.0" style="-fx-
background-color: #000000;" text="Отправить"
textFill="WHITE">
    <font>
        <Font name="System Bold" size="14.0" />
    </font></Button>
</VBox>
```