**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 Токун 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 |  | Токун 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](#_Toc140026179)

[1. ТРЕБОВАНИЯ К ПРОГРАММЕ 6](#_Toc140026180)

[1.1. Исходные Требования к программе 6](#_Toc140026181)

[1.1.1. Формальная постановка задачи 6](#_Toc140026182)

[1.1.2. Описание интерфейса 6](#_Toc140026183)

[1.1.3. Формат входных и выходных данных 8](#_Toc140026184)

[1.2. Уточнения требований после сдачи 1-ой версии 9](#_Toc140026185)

[1.3. Уточнения требований после сдачи 2-ой версии 9](#_Toc140026186)

[2. ПЛАН РАЗРАБОТКИ И РАСПРЕДЕЛЕНИЕ РОЛЕЙ В БРИГАДЕ 10](#_Toc140026187)

[2.1. План разработки 10](#_Toc140026188)

[2.2. Распределение ролей в бригаде 10](#_Toc140026189)

[3. ОСОБЕННОСТИ РЕАЛИЗАЦИИ 12](#_Toc140026190)

[3.1. Структуры данных 12](#_Toc140026191)

[3.2. Основные методы 14](#_Toc140026193)

[4. ТЕСТИРОВАНИЕ 20](#_Toc140026194)

[4.1. Тестирование основных функций продукта 20](#_Toc140026195)

[4.2. Тестирование граничных условий 22](#_Toc140026196)

[4.3. Тестирование интерфейса 25](#_Toc140026197)

[4.4. Тестирование структуры данных 29](#_Toc140026198)

[ЗАКЛЮЧЕНИЕ 33](#_Toc140026199)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 34](#_Toc140026200)

# **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

# **ТРЕБОВАНИЯ К ПРОГРАММЕ**

**1.1. Исходные Требования к программе**

* + 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. **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 состояние 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. **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)

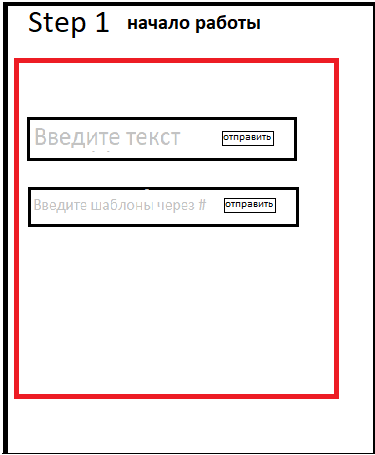


Рис. 1 – примерная иллюстрация открывшегося приложения

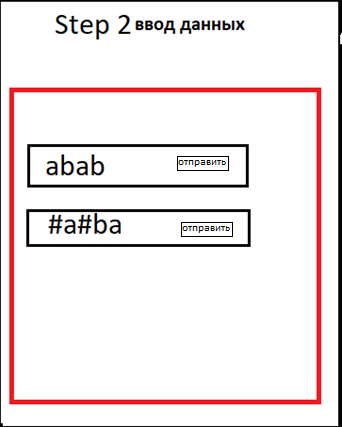


Рис. 2 – ввод данных для построения бора и нахождения подстрок в строке

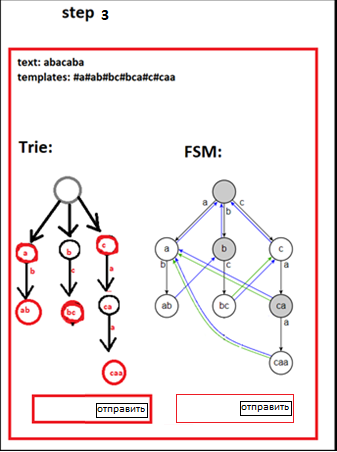


Рис. 3 – иллюстрация вывода

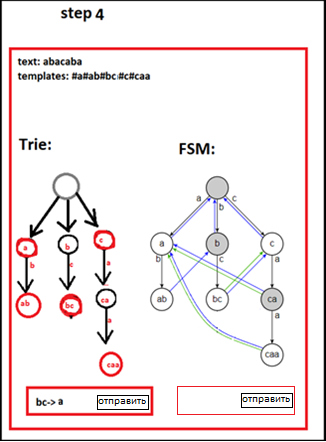


Рис. 4 – иллюстрация записи добавления вершины в бор

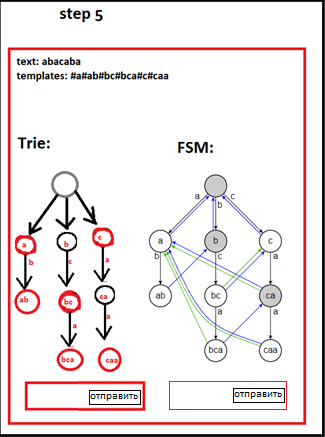


Рис. 5 – иллюстрация изменённого автомата и бора после добавления ребра

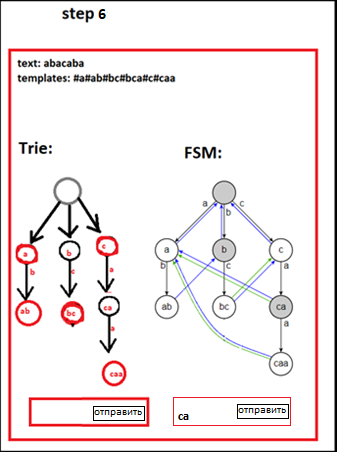


Рис. 6 – иллюстрация записи изменения статуса состояния

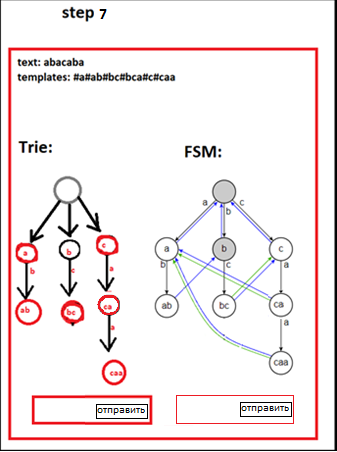


Рис. 7 – иллюстрация изменения бора и автомата

* + 1. **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.

## Уточнения требований после сдачи 1-ой версии

Добавить визуализацию  
Добавить удаление вершин

## 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

Токун – 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

сhildren: 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

## 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

back() – смена окна на начальное

1. **Тестирования**
   1. **Тестирование основных функций продукта**

* Тест на текст и один шаблон:
* Входные данные: текст и шаблон.
* Ожидаемый результат: алгоритм должен вернуть вхождения подстроки в текст и вывести визуализацию бора и конечного автомата.
* Тест на текст, в котором несколько шаблонов:
* Входные данные: текст и набор шаблонов написание которых разделяется “#”.
* Ожидаемый результат: алгоритм должен вернуть вывести визуализацию бора и конечного автомата.
* Тест на длинный текст вершин и малое количество шаблонов:
* Входные данные: текст и шаблон.
* Ожидаемый результат: алгоритм должен вернуть вхождения подстроки в текст и вывести визуализацию бора и конечного автомата.
* Тест на короткий текст вершин и большое количество шаблонов:
* Входные данные: текст и шаблон.
* Ожидаемый результат: алгоритм должен вернуть вхождения подстроки в текст и вывести визуализацию бора и конечного автомата.
* Тест на длинный текст и большое количесвто шаблонов:
* Входные данные: текст и шаблон.
* Ожидаемый результат: алгоритм должен вернуть вхождения подстроки в текст и вывести визуализацию бора и конечного автомата.
* **Тестирование граничных условий**
* Тест на пустой текст:
* Входные данные: пустой текст.
* Ожидаемый результат: вывод предупреждения на экране об отсутвии текста.
* Тест на отсутствие шаблонов:
* Входные данные: непустой текст и 0 шаблонов.
* Ожидаемый результат: вывод предупреждения об отсутствии шаблонов.
* Тест на повторяющиеся шаблоны:
* Входные данные: текст и набор шаблонов написание которых разделяется “#” и при этом некоторые одинаковые.
* Ожидаемый результат: алгоритм должен вернуть вхождения подстроки в текст исключив лишние шаблоны и вывести визуализацию бора и конечного автомата.
* Тест на то что один из шаблонов больше текста:
* Входные данные: текст и набор шаблонов написание которых разделяется “#” и при один из них больше текста.
* Ожидаемый результат: алгоритм должен вернуть вхождения подстроки в текст исключив большие шаблоны и вывести визуализацию бора и конечного автомата без учёта этого шаблона.
  1. **Тестирование интерфейса**
* Тестирование интерфейса:
* Входные данные: текст и набор шаблонов написание которых разделяется “#”.
* Ожидаемый результат: алгоритм должен вернуть вывести визуализацию бора и конечного автомата.
* Тест на случай отсутствия текста:
* Входные данные: пустой текст.
* Ожидаемый результат: вывод предупреждения на экране об отсутвии текста.
* Тест на случай некорректных входных данных:
* Входные данные: некорректный текст и набор некорректных шаблонов написание которых разделяется “#”.
* Ожидаемый результат: алгоритм должен обработать данную ситуацию и вернуть сообщение об ошибке или пустой результат.
* **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.
  1. **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, 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

1. Kotlin docs. URL:

<https://kotlinlang.org/docs/home.html>.

1. JavaFX DOCUMENTATION. URL:

<https://openjfx.io/>.

1. Курс введение в Kotlin JVM. URL:

<https://stepik.org/course/5448/syllabus> .

**приложение А**

**КОД программы**

Файл Main.kt

package com.example.ahocorasick

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("hello-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.ahocorasick

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.Image

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 с

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!!

}

/\*\*

\* Возваращет 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 суффискной 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

activateAlg()

} else if (txt.getText().isEmpty()) wrong.text = "there is no text"

else if (patterns.getText().isEmpty()) wrong.text = "there is no patterns"

}

class State(val name: String, val transitions: MutableMap<Char, State> = mutableMapOf())

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)

}

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.toString()))

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 buildTrie(patterns: List<String>): State {

// val root = State(" ")

//

// for (pattern in patterns) {

// var currentState = root

//

// val patternChars = pattern.toCharArray()

// for (i in patternChars.indices) {

// val char = patternChars[i]

// if (!currentState.transitions.containsKey(char)) {

// val newState = State("${currentState.name}${char}")

// currentState.transitions[char] = newState

// }

//

// currentState = currentState.transitions[char]!!

// }

// }

//

// return root

// }

//

// fun saveTrieAsPNG(automaton: State, filename: String, path: String = "") {

// val graph = Factory.mutGraph().setDirected(true)

// val nodes = mutableMapOf<State, MutableNode>()

//

// fun buildGraph(state: State) {

// val mutableNode = nodes.getOrPut(state) { Factory.mutNode(state.name) }

// graph.add(mutableNode)

//

// for (transition in state.transitions.values) {

// buildGraph(transition)

//

// val transitionNode = nodes[transition]!!

// mutableNode.addLink(Factory.to(transitionNode))

// }

// }

//

// 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}")

// }

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) {

val graph = Factory.mutGraph().setDirected(true)

val nodes = mutableMapOf<AutomatonNode, MutableNode>()

fun buildGraph(node: AutomatonNode) {

val mutableNode = nodes.getOrPut(node) { Factory.mutNode(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.toString()))

mutableNode.addLink(link)

}

if (node.outputLink != null) {

val outputLinkNode = nodes[node.outputLink!!]!!

val outputLinkLink = Factory.to(outputLinkNode).with(Label.of(" ")).add(Color.BLUE)

mutableNode.addLink(outputLinkLink)

}

if (node.suffixLink != null) {

val suffixLinkNode = nodes[node.suffixLink!!]!!

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 relativePath = "src/main/resources/com/example/ahocorasick" // Относительный путь от текущей директории

val file =

File(File(System.getProperty("user.dir"), relativePath).absolutePath)

val renderedGraph = Graphviz.fromString(dot).render(Format.PNG).toFile(file)

println("Автомат сохранен в файл: $filename")

}

fun buildGraph(patterns: String): guru.nidi.graphviz.model.MutableGraph {

val graph = Factory.mutGraph("example").setDirected(true)

val nodes = patterns.split("#")

for (i in nodes.indices) {

val node = Factory.node(nodes[i])

graph.add(node)

if (i > 0) {

val prevNode = Factory.node(nodes[i - 1])

graph.add(prevNode.link(node).with(Color.RED))

}

}

return graph

}

private fun makePic() {

val patterns = "${patterns.text}"

val graph = buildGraph(patterns)

val automaton1 = buildAutomat(patterns.split("#"))

saveAutomatonAsPNG(automaton1, "fsm.png")

val dot = Graphviz.fromGraph(graph).render(Format.DOT).toString()

val patterns1 = patterns.split("#")

val automaton2 = buildAutomaton(patterns1)

markTrueNodes(automaton2)

val relativePath = "src/main/resources/com/example/ahocorasick" // Относительный путь от текущей директории

val path1 = File(System.getProperty("user.dir"), relativePath).absolutePath

val filename1 = "trie.png"

//val path1 = "C:/Users/LENOVO/Documents/GitHub/educational-practice/AhoCorasick/src/main/resources/com/example/ahocorasick/"

saveTreAsPNG(automaton2, filename1, path1)

}

@FXML

private fun activateAlg() {

textText.text = "Text:"

patternsText.text = "Patterns: ${patterns.text}"

val bor = Bor()

for (word in patterns.text.split('#')) {

bor.insert(word)

}

print(bor.toString(tab = 0))

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

makePic()

}

// val filename = "trie.png"

// val relativePath = "../../../../resources/com/example/ahocorasick" // Относительный путь от текущей директории

//

// val path = File(System.getProperty("user.dir"), relativePath).absolutePath

//

// saveTrieAsPNG(automaton, filename, path)

//

// fsmImg.image = get

//

// }

//

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}" }}"

}

@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}" }}"

}

}

@FXML

private fun goodSwitchTerminal() {

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}" }}"

}

@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

}

}

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.ahocorasick.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>

<Font name="Bell MT" size="16.0" />

</font>

</Text>

</children>

</GridPane>

<Button fx:id="switchTextButton" alignment="CENTER" contentDisplay="CENTER" mnemonicParsing="false" onAction="#switchText" style="-fx-background-color: #000000;" text="поменять" textAlignment="CENTER" textFill="WHITE" visible="false" GridPane.halignment="CENTER" GridPane.rowIndex="1" GridPane.valignment="CENTER" />

</children>

</GridPane>

<GridPane prefHeight="280.0" prefWidth="377.0">

<columnConstraints>

<ColumnConstraints hgrow="SOMETIMES" maxWidth="220.0" minWidth="10.0" prefWidth="220.0" />

<ColumnConstraints hgrow="SOMETIMES" maxWidth="220.0" minWidth="10.0" prefWidth="220.0" />

<ColumnConstraints hgrow="SOMETIMES" maxWidth="178.0" minWidth="10.0" prefWidth="115.0" />

</columnConstraints>

<rowConstraints>

<RowConstraints maxHeight="191.0" minHeight="10.0" prefHeight="190.0" vgrow="SOMETIMES" />

<RowConstraints maxHeight="77.0" minHeight="0.0" prefHeight="50.0" vgrow="SOMETIMES" />

<RowConstraints maxHeight="97.0" minHeight="10.0" prefHeight="29.0" vgrow="SOMETIMES" />

</rowConstraints>

<children>

<ImageView fx:id="trieImg" fitHeight="192.0" fitWidth="277.0" pickOnBounds="true" preserveRatio="true" visible="false">

<image>

<Image url="@fsm.png" />

</image></ImageView>

<ImageView fx:id="fsmImg" fitHeight="189.0" fitWidth="275.0" pickOnBounds="true" preserveRatio="true" visible="false" GridPane.columnIndex="1">

<image>

<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="switch terminal of" GridPane.columnIndex="2" GridPane.rowIndex="1" />

<Button fx:id="delButton" mnemonicParsing="false" onAction="#delNode" text="Button" 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>