Федеральное агентство связи

Федеральное государственное бюджетное образовательное учреждение

высшего образования

«Сибирский государственный университет

телекоммуникаций и информатики»

**Курсовая работа**

**Вариант 1**

Выполнил:

Студент 4 курса

ИВТ, гр. ИП-712

Алексеев С.В.

Проверил:

Доцент кафедры ПМИК

Бах О.А.

Новосибирск 2020 г.

Оглавление

[Задание 2](#_Toc59125013)

[Описание алгоритма 2](#_Toc59125014)

[Листинг 2](#_Toc59125015)

# Задание

Написать программу, которая по предложенному описанию языка построит детерминированный конечный автомат, распознающий этот язык, и проверит вводимые с клавиатуры цепочки на их принадлежность языку. Предусмотреть возможность поэтапного отображения на экране процесса проверки цепочек. Функция переходов ДКА может изображаться в виде таблицы и графа (выбор вида отображения посредством меню). Вариант задания языка: (1) Алфавит, обязательная конечная подцепочка всех цепочек языка и кратность вхождения выбранного символа алфавита в любую цепочку языка.

Отчёт должен содержать:

1) титульный лист;

2) номер варианта и текст задания;

3) описание алгоритма решения задачи с иллюстрацией на примере;

4) описание основных блоков программы;

5) распечатку текста программы;

6) результаты тестирования.

Вместе с отчётом необходимо сдать преподавателю электронный вариант программы – все исходники, exe-модуль, файлы данных, файлы результатов.

# Описание алгоритма

Задание несколько изменено преподавателем: кратность считается только для символов, не входящих в конечную подцепочку, т.е. если в конечной подцепочке будут символы для кратности, то граф получится неправильным.

Для рисования графа использовал библиотеку MSAGL на языке C#.

Все шаги описаны и прокомментированы в коде.

Сначала расписал построение графов для простейших случаев с пустой конечной подцепочкой.

Если подцепочка не пустая и кратность больше одного, то смотрю на кратность и символ кратности, считаю число нужных состояний для приёма символов кратности. Строю состояния для приёма символов кратности.

Строю возврат в самих себя состояний по алфавиту без символа кратности при проходе по состояниям, принимающим символы кратности, не считая нулевого состояния, т.к. из него переходы могут быть к приёму конечной подцепочки, т.к. кратность чился символов для кратности там подходящая( м.б. ноль штук, а может быть кратно заданному пользователем числу).

Строю состояния для приёма конечной подцепочки из нулевого состояния.

В каждом из состояний, принимающих конечную подцепочку, строю префикс(вся считанная до текущего состояния «корректная» часть подцепочки) и суффиксы – возможные подцепочки при теоретическом приёме следующего символа в текущем состоянии. Перебираю все префиксы, начиная с префикса длины один и т.д., наращивая префикс до префикса, считанного в текущем состоянии. Сравниваю эти префиксы с теоретическими суффиксами(также сначала длины один(теоретически принимаемый данным состоянием символ), а потом наращиваю символы, читая их из «корректной» считанной подцепочки справа налево и нахожу совпадения. Как только нашёл равные суффикс и префикс, так сразу запоминаю эту комбинацию(подцепочка, состояние, в котором такая подцепочка считана). При переборе всех вариантов суффиксов/префиксов для данного состояния, нахожу равные суффикс и префикс максимальной длины. Строю ребро графа из текущего состояния в состояние, где был считан префикс максимальной длины. Если этот префикс длины один, то, естетсвенно, ребро строится в нулевое состояние.

Начальное состояние всегда q0. Конечное – либо q0, если коненчая подцепочка пустая, иначе конечное - последнее состояние для приёма последнего символа конечной подцепочки.

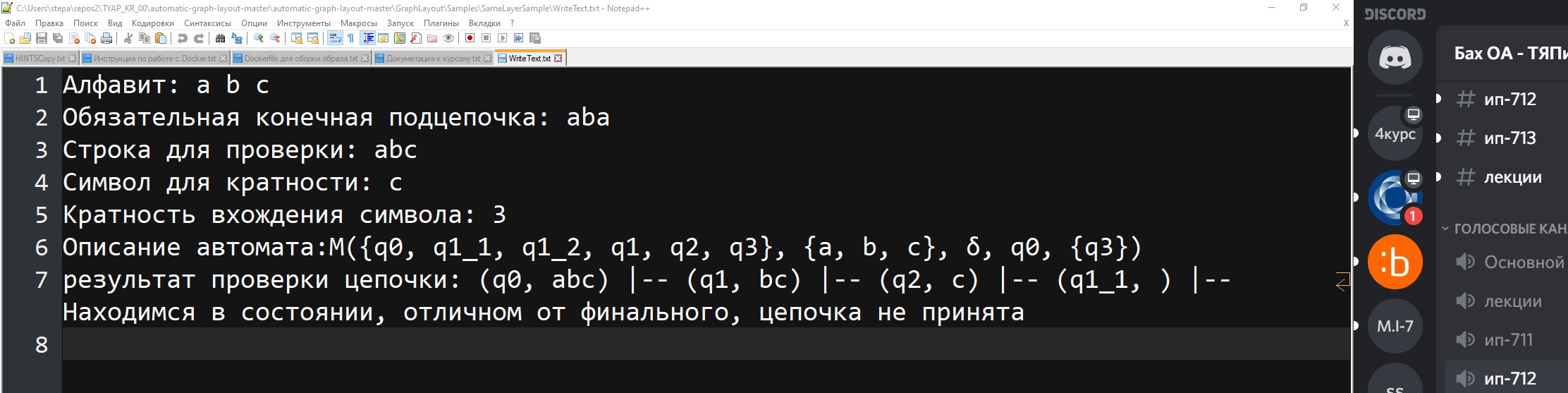
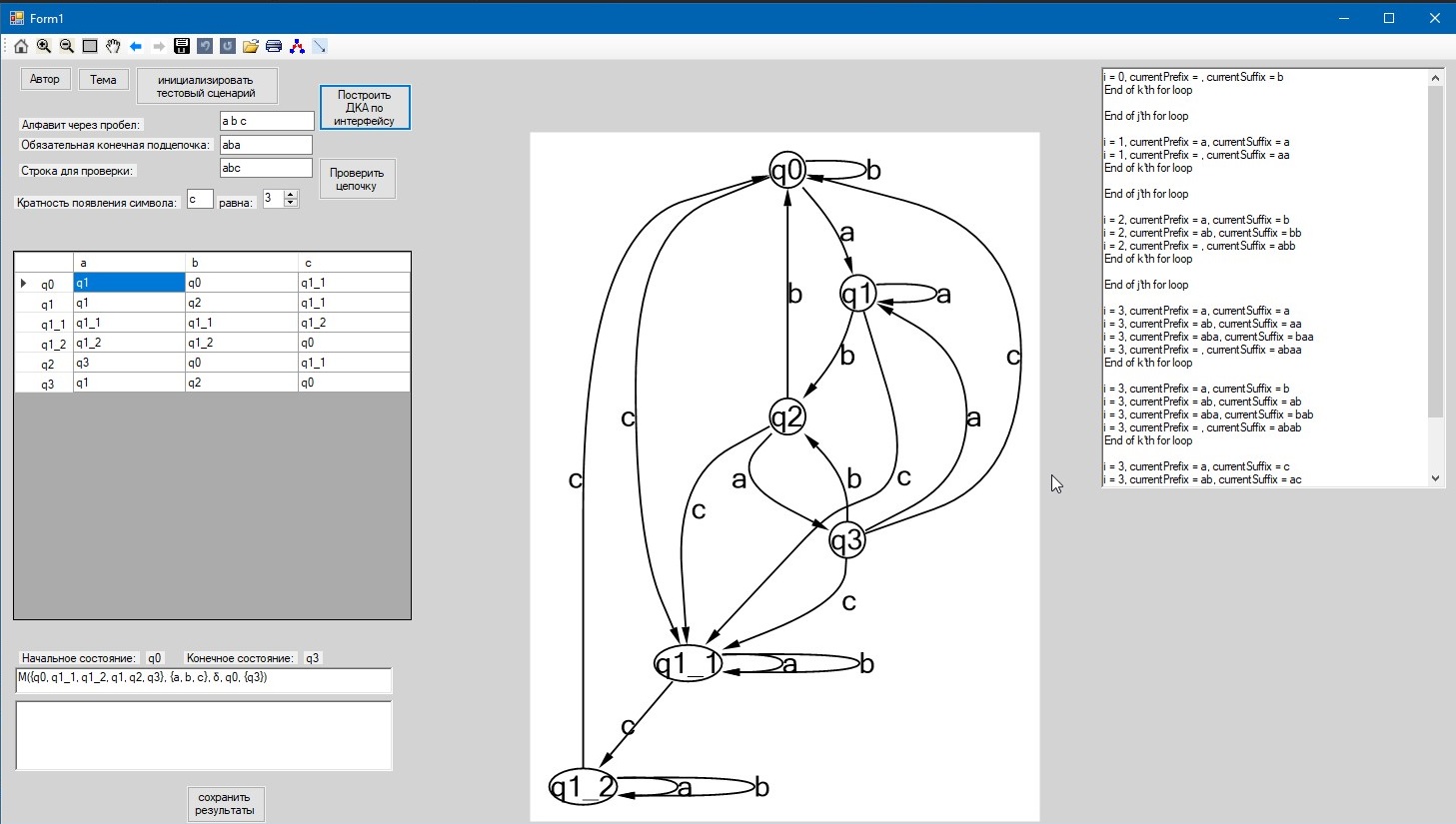
В итоге имеем граф, соотвтетствующий требованиям. По нему строю таблицу переходов для отображения в datagridview.

Остальные действия тривиальны:

Проверка подцепочки проходит переходом от текущего состояния к следующему по соответствующему символу алфавита с помощью построенной таблицы переходов. В итоге смотрю на текущее состояние при прочтении всей цепочки. Если состояние конечное – цепочка принята.

Сохранение результатов в файл и скриншот производятся соотвтетствующей кнопкой. И т.д.

# Скриншоты



# Листинг

using System;

using System.Collections;

using System.Collections.Generic;

using System.Drawing;

using System.Drawing.Imaging;

using System.Globalization;

using System.Linq;

using System.Threading;

using System.Windows.Forms;

using Microsoft.Msagl.Drawing;

using Microsoft.Msagl.GraphViewerGdi;

using Microsoft.Msagl.Layout.Layered;

namespace SameLayerSample {

public partial class Form1 : Form {

public Form1() {

InitializeComponent();

formForTheInputs = this;

}

public string stringToCheck = "";

public string finalSubstring = "";

public string[] finalSubstringInArrayOfStrings;

public string symbolForMultiplicity = "";

int multiplicity;

public string[] alphabet;

//public int kratnost = 0;

Form1 formForTheInputs;

public string[] stringTocheckInArrayOfStrings;

string initialState = "q0";

string finalState = "";

string currentState = "";

Graph graph;

IEnumerable<string> alphabetInIEnumerable;

public bool finalSubstringContainsSymbolForMultiplicity = false;

public bool mulSymbIsTheFirstSymbOfFSSg = false;// multiplicity symbol is the first symbol of the FinalSubstring

int numEqualInBgnFSSg = 0;//numberOfEqualSymbolsInTheBeginningOfFinalSubstring

int numMulSymbInFSSg = 0;//numberOfSymbolsForMultiplicityInFinalSubstring

int indexOfTheLastMul = 0;//Индекс последнего символа кратности в конечной подстроке

int indexOfTheFirstMul = 0;//Индекс первого символа кратности в конечной подстроке

List<Node> list;

string fullPrefix = "";

string suffix = "";

string currentPrefix = "";

string currentSuffix = "";

bool foundEqualSuffAndPref = false;

int lengthOfEqualSufAndPref = 0;

string stateFrom = "";

string stateTo = "";

string symbOfEqualSufAndPref = "";

string currentStateName = "";

string newStateName = "";

int numMulSymbs = 0;

string[] statesInStrings;

public List<string> lst;

public void initializeTestCase\_abcdefg\_3a\_bfg\_() {

textBox1StringToCheck.Text = "abc";

textBox2Alphabet.Text = "a b c";

textBox3FinalSubString.Text = "aba";

textBox4SymbolForMultiplicity.Text = "c";

numericUpDown1Multiplicity.Value = 3;

lst = new List<string>();

lst.Add("rt");

lst.Add("ry");

lst.Add("rt");

var lst2 = lst.Distinct();

foreach (var item in lst2) {

richTextBox1Helper.AppendText("item = " + item + ", ");

}

}

GViewer gViewer;

string alphabetInString = "";

public string setExceptParameter(string set, string str) {//returns a string, for example:

//set.Except(str) or {a, b, c}/b = {a, c} = "a c". I.e. the returned string is "a c".

string[] aa = new string[1];// НЕ ИСПОЛЬЗОВАТЬ ЭТОТ МЕТОД

aa[0] = str;

string[] setInArray = set.Split(' ');

IEnumerable<string> setWithout\_str = setInArray.Except(aa);

string setWithout\_a\_InString = "";

foreach (object v in setWithout\_str) {

setWithout\_a\_InString += v + " ";

}

return setWithout\_a\_InString;

}

public void buildDFASuffPref() {

richTextBox1Helper.Clear();

//statesInStrings = n

lst = new List<string>();

Thread.CurrentThread.CurrentCulture = CultureInfo.InvariantCulture;//turning exception messages to English

Thread.CurrentThread.CurrentUICulture = CultureInfo.InvariantCulture;

numEqualInBgnFSSg = 0;

numMulSymbInFSSg = 0;

if (gViewer != null) {

gViewer.Dispose();

}

gViewer = new GViewer() { Dock = DockStyle.Fill };

SuspendLayout();

Controls.Add(gViewer);

ResumeLayout();

graph = new Graph();

var sugiyamaSettings = (SugiyamaLayoutSettings)graph.LayoutAlgorithmSettings;

sugiyamaSettings.NodeSeparation \*= 2;

//сначала надо построить базу: состояния для считывания символов кратности,

//состояния для считывания конечной подстроки: это всё пока для кратности равной 1... с кратностью больше 1, вероятно, надо будет

//строить по-другому

if (finalSubstring.Length == 0 && multiplicity == 1) {// I

//graph.AddEdge("q0", "q0").LabelText = alphabetInString;

for (int i = 0; i < alphabet.Length; i++) {

graph.AddEdge("q0", alphabet[i], "q0");

}

label8InitialState.Text = "q0";

label9FinalState.Text = "q0";

lst.Add("q0");

}

else if (finalSubstring.Length == 0 && multiplicity > 1) {// II

string[] sethWithSymbForMul = { symbolForMultiplicity };

string[] alphWOSymbForMul = (string[])alphabet.Except(sethWithSymbForMul).ToArray();

for (int i = 0; i < multiplicity - 1; i++) {

graph.AddEdge("q" + i, "q" + (i + 1)).LabelText = symbolForMultiplicity;

for (int j = 0; j < alphWOSymbForMul.Length; j++) {

graph.AddEdge("q" + i, alphWOSymbForMul[j], "q" + i);

}

// graph.AddEdge("q" + i, "q" + i).LabelText = setExceptParameter(alphabetInString, symbolForMultiplicity);

lst.Add("q" + i);

lst.Add("q" + (i + 1));

}

graph.AddEdge("q" + (multiplicity - 1), "q" + 0).LabelText = symbolForMultiplicity;

//graph.AddEdge("q" + (multiplicity - 1), "q" + (multiplicity - 1)).LabelText = setExceptParameter(alphabetInString, symbolForMultiplicity);

for (int j = 0; j < alphWOSymbForMul.Length; j++) {

graph.AddEdge("q" + (multiplicity - 1), alphWOSymbForMul[j], "q" + (multiplicity - 1));

}

label8InitialState.Text = "q0";

label9FinalState.Text = "q0";

lst.Add("q" + (multiplicity - 1));

}

else {

list = new List<Node>();//Создал список состояний, в которых буду хранить суффиксы, префиксы и другое в удобном для меня виде

int numOfAdditEdges = 0;//numberOfSymbolsForMultiplicityToCreateAdditionalStates better to say "additional edges"

if (!(multiplicity == 1)) {//Если кратность больше 1, то надо добавить состояния для приёма символов кратности

numOfAdditEdges = 1;

while (((multiplicity + numOfAdditEdges) % multiplicity) != 0) {//Подбираю число дополнительных рёбер

numOfAdditEdges++;

}

//Строю состояния для приёма символов кратности:

string[] sethWithSymbForMul = { symbolForMultiplicity };

string[] alphWOSymbForMul = (string[])alphabet.Except(sethWithSymbForMul).ToArray();

for (int i = 0; i < numOfAdditEdges - 1; i++) {

if (i == 0)//Если строю из нулевого состояния, то нулевое в прежнем виде, а добавочные в измененном

{

graph.AddEdge("q" + i, symbolForMultiplicity, "q1\_" + (i + 1));

lst.Add("q" + i);

lst.Add("q1\_" + (i + 1));

for (int p = 0; p < alphWOSymbForMul.Length; p++) {//Закольцовываю на себя состояния по всем символам кроме кратности

graph.AddEdge("q1\_" + (i + 1), alphWOSymbForMul[p], "q1\_" + (i + 1));

lst.Add("q1\_" + (i + 1));

}

}

else {

graph.AddEdge("q1\_" + i, symbolForMultiplicity, "q1\_" + (i + 1));

lst.Add("q1\_" + i);

lst.Add("q1\_" + (i + 1));

for (int p = 0; p < alphWOSymbForMul.Length; p++) {//Закольцовываю на себя состояния по всем символам кроме кратности

graph.AddEdge("q1\_" + (i + 1), alphWOSymbForMul[p], "q1\_" + (i + 1));

}

}

//list.Add(new Node("q" + i, prefix, suffix));

}//Добавляю переход из последнего символа для кратности в нулевое состояние:

graph.AddEdge("q1\_" + (numOfAdditEdges - 1), symbolForMultiplicity, "q0").Attr.Id =

("q1\_" + (numOfAdditEdges - 1) + " " + symbolForMultiplicity + " " + "q0");

lst.Add("q1\_" + (numOfAdditEdges - 1));

}

// if (multiplicity == 1)

{ //Строю состояния для конечной подстроки:

for (int i = 0; i < finalSubstring.Length; i++) {

currentStateName = "q" + (0 + i);//was: currentStateName = "q" + (numOfAdditEdges + i);

newStateName = "q" + (0 + i + 1);//was: newStateName = "q" + (numOfAdditEdges + i + 1);

graph.AddEdge(currentStateName, finalSubstringInArrayOfStrings[i], newStateName).Attr.Id =

(currentStateName + " " + finalSubstringInArrayOfStrings[i] + " " + "q" + (i + 1));

lst.Add(currentStateName);

lst.Add("q" + (i + 1));

lst.Add(newStateName);

fullPrefix = (finalSubstring.Substring(0, i));//it's full from the beginning up to i'th symbol

list.Add(new Node(currentStateName, fullPrefix));//создал свой узел с префиксом и именем как у оригинального состояния

//string[] strToExcept = { finalSubstringInArrayOfStrings[i] };

string[] alphWithOneCorrSymb = { finalSubstringInArrayOfStrings[i] };

string[] alphWOCorSymb = (string[])alphabet.Except(alphWithOneCorrSymb).ToArray();

if (multiplicity > 1) {//Если кратность больше одного, то нужно не допустить переходов по символу кратности в q0

//Если кратность больше одного, то надо из каждого состояния, читающего конечную подцепочку

//по символу кратности переходить в состояние q1\_1. А переходы по этому символу в нулевое, соотвтетсвенно отменить.

string[] alphWithSymbForMul = { symbolForMultiplicity };

alphWOCorSymb = (string[])alphWOCorSymb.Except(alphWithSymbForMul).ToArray();

if (i != 0)//Из нулевого в q1\_1 уже есть ребро, поэтому второе добавлять не нужно

{

graph.AddEdge(currentStateName, symbolForMultiplicity, "q1\_1");

lst.Add(currentStateName);

lst.Add("q1\_1");

}

}

//Теперь надо найти равные суффикс и префикс максимальной длины и сделать переход по соотвтетствующему символу:

currentSuffix = "";

for (int j = 0; j < alphWOCorSymb.Length; j++) {//Здесь вообще в алфавите один символ b остался...

//сначала иду по алфавиту. Для каждого символа алфавита д.б. переход(построение ребра).

for (int k = fullPrefix.Length - 1 + 1; k >= 0; k--) {//-1+1 Написано для ясности, чтобы понять, что мне нужен один индекс, по которому я не

//буду брать часть префикса для построения суффикса, а возьму только символ алфавита. +1 Нужен для одной дополнительной итерации.

//потом иду по длине префикса, т.к. суффикс и префикс д.б. одной длины

//Каждой длине префикса, видимо, соостветствует только одно состояние. Так что здесь же можно и состояния сразу перебирать

//Состоянию с индексом k соотвтетствует... Нет. Префикс длины 0 у нас в нулевом состоянии, принимающем первый

//символ КПС(конечной подстроки), и имеющем переход в следующее состояние по этому символу. Это число (numOfAdditEdges + k).

//Длина префикса ноль, длина суффикса 1. Сравнение на нулевом(numOfAdditEdges + 0) состоянии вернёт ложь всегда. Единственный переход

//будет по следующему верному символу кпс. Все сотальные символы закольцуют нулевое состояние на само себя.

//Вообще все состояния, не имеющие суффикса равного префиксу должны переходить в нулевое состояние по соотвтетствующему символу

//подцепочки. Префикс имеет смысл рассматривать только с начала, по очереди сравнивая все возможные суффиксы длины префикса с этими

//же префиксами. Т.е. ещё один цикл д.б. по всем вариантам суффиксов... Нет. Этот перебор уже идёт в цикле с индексом j.

//Итак, имею следующий символ и длину префикса. На следующем шаге текущего цикла изменится длина префикса. Как сравнивать с префиксом

//нулевой длины? Вообще получение суффикса:

currentSuffix = "";

if (!(k == fullPrefix.Length)) {//если не находимся в дополнительной итерации, то рассматриваем суффиксы длины 2 и больше.

//Индекс для взятия части префикса уменьшается от максимального, а число забираемых символов соотвтетсвенно увеличивается:

currentSuffix += fullPrefix.Substring(k, (fullPrefix.Length) - k) + alphWOCorSymb[j];

}

else {//Иначе берём только символ алфавита.

currentSuffix = alphWOCorSymb[j];

}

//На этом моменте понятно, что я смогу перебрать все суффиксы. Но смогу ли я перебрать все префиксы без доп. цикла...

//Префиксы мне нужны будут только длины равной длине суффикса. При создании всех предыдущих состояний, читающих

//кпс, я также проходил по всем предыдущим состояниям... Так что итерация по всем префиксам кажется нужной...

//Хотя нет. Достаточно одного назначения:

// stateFrom = "q" + i;

// stateTo = "q0";

currentPrefix = "";

if (!(fullPrefix.Length == 0) && (fullPrefix.Length >= currentSuffix.Length))

currentPrefix = fullPrefix.Substring(0, currentSuffix.Length);

richTextBox1Helper.AppendText("i = " + i + ", currentPrefix = " + currentPrefix + ", currentSuffix = " + currentSuffix + "\n");

if (currentPrefix.Equals(currentSuffix)) {

foundEqualSuffAndPref = true;

lengthOfEqualSufAndPref = currentPrefix.Length;

//stateFrom = "q" + i;

stateFrom = currentStateName;

stateTo = "q" + currentPrefix.Length;//Если обнаружится подходящий суффикс большей длины, то он обязательно будет переназначен,

//т.к. идём от меньшей длины суффикса к большей по k

symbOfEqualSufAndPref = alphWOCorSymb[j];

}

else {

if (k == 0 && !foundEqualSuffAndPref) {

graph.AddEdge(currentStateName, alphWOCorSymb[j], "q0").Attr.Id = (currentStateName + " " + alphWOCorSymb[j] + " " + "q0");

lst.Add(currentStateName);

lst.Add("q0");

}

}

}//Проверили суффиксы всех длин и теперь можно добавить ребро, если нашли суффикс и префикс максимальной длины:

richTextBox1Helper.AppendText("End of k'th for loop\n\n");

if (foundEqualSuffAndPref) {

foundEqualSuffAndPref = false;

graph.AddEdge(stateFrom, symbOfEqualSufAndPref, stateTo).Attr.Id = (stateFrom + " " + symbOfEqualSufAndPref + " " + stateTo);

lst.Add(stateFrom);

lst.Add(stateTo);

}

}

richTextBox1Helper.AppendText("End of j'th for loop\n\n");

}//Здесь нужно написать добавление рёбер из последнего состояния:

string numOfAdditEdgesPlusFSSgLength = (0 + finalSubstring.Length).ToString();//здесь косяк... нет. Здесь как раз переход правильный в q1\_1

currentStateName = "q" + numOfAdditEdgesPlusFSSgLength;

//newStateName = "q" + (numOfAdditEdges + i + 1);

if (multiplicity > 1) {//Если кратность больше одного, то нужно не допустить переходов по символу кратности в q0

//Если кратность больше одного, то надо из каждого состояния, читающего конечную подцепочку

//по символу кратности переходить в состояние q1\_1. А переходы по этому символу в нулевое, соотвтетсвенно отменить.

//string[] alphWithSymbForMul = { symbolForMultiplicity };

// alphWOCorSymb = (string[])alphWOCorSymb.Except(alphWithSymbForMul).ToArray();

graph.AddEdge(currentStateName, symbolForMultiplicity, "q1\_1");

lst.Add(currentStateName);

lst.Add("q1\_1");

}

fullPrefix = (finalSubstring.Substring(0, finalSubstring.Length));//it's full from the beginning up to i'th symbol

for (int j = 0; j < alphabet.Length; j++) {

for (int k = fullPrefix.Length - 1 + 1; k >= 0; k--) {

currentSuffix = "";

if (!(k == fullPrefix.Length)) {//если не находимся в дополнительной итерации, то рассматриваем суффиксы длины 2 и больше.

//Индекс для взятия части префикса уменьшается от максимального, а число забираемых символов соотвтетсвенно увеличивается:

currentSuffix += fullPrefix.Substring(k, (fullPrefix.Length) - k) + alphabet[j];

}

else {//Иначе берём только символ алфавита.

currentSuffix = alphabet[j];

}

//На этом моменте понятно, что я смогу перебрать все суффиксы. Но смогу ли я перебрать все префиксы без доп. цикла...

//Префиксы мне нужны будут только длины равной длине суффикса. При создании всех предыдущих состояний, читающих

//кпс, я также проходил по всем предыдущим состояниям... Так что итерация по всем префиксам кажется нужной...

//Хотя нет. Достаточно одного назначения:

// stateFrom = "q" + i;

// stateTo = "q0";

currentPrefix = "";

if (!(fullPrefix.Length == 0) && (fullPrefix.Length >= currentSuffix.Length))

currentPrefix = fullPrefix.Substring(0, currentSuffix.Length);

richTextBox1Helper.AppendText("i = " + numOfAdditEdgesPlusFSSgLength +

", currentPrefix = " + currentPrefix + ", currentSuffix = " + currentSuffix + "\n");

if (currentPrefix.Equals(currentSuffix)) {

foundEqualSuffAndPref = true;

lengthOfEqualSufAndPref = currentPrefix.Length;

stateFrom = "q" + numOfAdditEdgesPlusFSSgLength;

stateTo = "q" + currentPrefix.Length;//Если обнаружится подходящий суффикс большей длины, то он обязательно

//будет переназначен,

//т.к. идём от меньшей длины суффикса к большей по k

symbOfEqualSufAndPref = alphabet[j];

}

else {

if (k == 0 && !foundEqualSuffAndPref && !(alphabet[j].Equals(symbolForMultiplicity))) {

graph.AddEdge("q" + numOfAdditEdgesPlusFSSgLength, alphabet[j], "q0").Attr.Id =

("q" + numOfAdditEdgesPlusFSSgLength + " " + alphabet[j] + " " + "q0");

lst.Add("q" + numOfAdditEdgesPlusFSSgLength);

lst.Add("q0");

}

}

}//Проверили суффиксы всех длин и теперь можно добавить ребро, если нашли суффикс и префикс максимальной длины:

richTextBox1Helper.AppendText("End of k'th for loop\n\n");

if (foundEqualSuffAndPref) {

foundEqualSuffAndPref = false;

graph.AddEdge(stateFrom, symbOfEqualSufAndPref, stateTo).Attr.Id = (stateFrom + " " + symbOfEqualSufAndPref + " " + stateTo);

lst.Add(stateFrom);

lst.Add(stateTo);

}

}

}

}

var lstDistinct = lst.Distinct();

richTextBox1M.Clear();

richTextBox1M.AppendText("M({");

foreach (var item in lstDistinct) {

richTextBox1M.AppendText(item + ", ");

}

richTextBox1M.AppendText("}, {");

foreach (var item in alphabet) {

richTextBox1M.AppendText(item + ", ");

}

richTextBox1M.AppendText("}, ");

richTextBox1M.AppendText("δ, q0, {q" + finalSubstring.Length + "})");

richTextBox1M.Text = richTextBox1M.Text.Replace(", }", "}");

label8InitialState.Text = "q0";

label9FinalState.Text = "q" + finalSubstring.Length;

graph.Attr.OptimizeLabelPositions = true;

graph.Attr.SimpleStretch = true;

gViewer.Graph = graph;

buildDataGridView1ByGraph();

}

public void buildDataGridView1ByGraph() {

//Каждому состоянию сооответствует одна строка таблицы:

dataGridView1.RowCount = graph.NodeCount;

//Столбцов д.б. столько, сколько символов в алфавите:

dataGridView1.ColumnCount = alphabet.Length;

for (int i = 0; i < alphabet.Length; i++) {

dataGridView1.Columns[i].HeaderCell.Value = alphabet[i];

}

int p = 0;

Hashtable hash = graph.NodeMap;

List<string> lst = new List<string>();//обычно Hashtable не отсортирована, поэтому пришлось загнать её в List для сортировки и создания таблицы

foreach (var key2 in hash.Keys) {

lst.Add(key2.ToString());

}

lst.Sort();

string lastNode = "";

foreach (var item in lst) {

dataGridView1.Rows[p++].HeaderCell.Value = item;

finalState = lastNode = item;

}

foreach (var node in graph.Nodes)

foreach (Edge edge in node.Edges) {//построение datagridview по graph

for (int i = 0; i < dataGridView1.Rows.Count; i++) {

for (int j = 0; j < dataGridView1.Columns.Count; j++) {

if (edge.Source.ToString().Equals(dataGridView1.Rows[i].HeaderCell.Value.ToString())) {

string lbl = edge.LabelText.ToString();

string hdr = dataGridView1.Columns[j].HeaderText;

if (lbl.CompareTo(hdr) == 0) {

dataGridView1.Rows[i].Cells[j].Value = edge.Target.ToString();

}

}

}

}

}

}

}

public void readInformationFromTheInterface() {

stringToCheck = textBox1StringToCheck.Text;

finalSubstring = textBox3FinalSubString.Text;

finalSubstringInArrayOfStrings = stringToArrayOfStrings(finalSubstring);

symbolForMultiplicity = textBox4SymbolForMultiplicity.Text;

if (symbolForMultiplicity.Length != 1) {

MessageBox.Show("Символ для кратности должен быть один");

textBox4SymbolForMultiplicity.Text = "";

}

numMulSymbs = multiplicity = (int)numericUpDown1Multiplicity.Value;

if (multiplicity < 1) {

MessageBox.Show("Кратность не м.б. меньше одного");

numericUpDown1Multiplicity.Value = 1;

}

alphabet = textBox2Alphabet.Text.Split(' ');

alphabetInString = textBox2Alphabet.Text;

stringTocheckInArrayOfStrings = stringToArrayOfStrings(stringToCheck);

initialState = textBoxInitialState.Text;

finalState = textBoxFinalState.Text;

}

public string[] stringToArrayOfStrings(string stringToCheck) {

char[] stringTocheckInChars = stringToCheck.ToCharArray();

string[] stringToCheckInStringArray = new string[stringTocheckInChars.Length];

for (int i = 0; i < stringTocheckInChars.Length; i++) {

stringToCheckInStringArray[i] = stringTocheckInChars[i].ToString();

}

return stringToCheckInStringArray;

}

private void button2\_Click(object sender, System.EventArgs e) {

initializeTestCase\_abcdefg\_3a\_bfg\_();

}

private void button1\_Click(object sender, System.EventArgs e) {

buildFirstDFA\_AndDatagridviewByIt();

}

private void button3\_Click(object sender, System.EventArgs e) {

checkAStringByDataGridview1();

}

private void заданиеToolStripMenuItem\_Click(object sender, EventArgs e) {

}

private void button4\_Click(object sender, EventArgs e) {

MessageBox.Show("Вариант № 1\n Написать программу, которая по предложенному описанию языка построит " +

"детерминированный конечный автомат, распознающий этот язык, " +

"и проверит вводимые с клавиатуры цепочки на их принадлежность языку. " +

"Предусмотреть возможность поэтапного отображения на экране " +

"процесса проверки цепочек. Функция переходов ДКА может изображаться в виде таблицы и " +

"графа(выбор вида отображения посредством меню). Вариант(первый) задания языка:\n\n" +

" Алфавит, обязательная конечная подцепочка всех цепочек языка " +

"и кратность вхождения выбранного символа алфавита в любую цепочку языка.");

}

private void button5\_Click(object sender, EventArgs e) {

MessageBox.Show("Алексеев Степан Владимрович. Группа ИП-712. Ноябрь 2020");

}

private void button6\_Click(object sender, EventArgs e) {

string outString = "";

outString += "Алфавит: " + alphabetInString + "\n" + "Обязательная конечная подцепочка: " + finalSubstring + "\n" + "Строка для проверки: " + stringToCheck + "\n" +

"Символ для кратности: " + symbolForMultiplicity + "\n" + "Кратность вхождения символа: " + multiplicity + "\n" +"Описание автомата:" +richTextBox1M.Text +

"\n"+ "результат проверки цепочки: " + richTextBox2CheckResults.Text;

System.IO.File.WriteAllText(@"C:\Users\stepa\repos2\TYAP\_KR\_00\automatic-graph-layout-master\automatic-graph-layout-master\GraphLayout\Samples\SameLayerSample\WriteText.txt", outString);

Rectangle bounds = Screen.GetBounds(Point.Empty);

using (Bitmap bitmap = new Bitmap(bounds.Width, bounds.Height)) {

using (Graphics g = Graphics.FromImage(bitmap)) {

g.CopyFromScreen(Point.Empty, Point.Empty, bounds.Size);

}

bitmap.Save("C:\\Users\\stepa\\repos2\\TYAP\_KR\_00\\automatic-graph-layout-master\\automatic-graph-layout-master\\GraphLayout\\Samples\\SameLayerSample\\My\_Img.jpg", ImageFormat.Jpeg);

}

}

private void button7\_Click(object sender, EventArgs e) {

readInformationFromTheInterface();

buildDFASuffPref();

}

private void button7\_Click0(object sender, EventArgs e) readInformationFromTheInterface();

if (finalSubstring.Contains(multiplicity.ToString())) {

}

else if (finalSubstring.Length != 0 && multiplicity != 0 && !finalSubstring.Contains(multiplicity.ToString())) {

BuildDFAFirstCase\_multiplicityNotEqualToZeroAndTheSymbolIsNotInTheFinalSubstring();

}

else if (finalSubstring.Length == 0) {

BuildDFAFinalSubstringIsEmpty\_case\_3();

}

}

}

}

class Node {

public Node(string name, string prefix) {

this.name = name;

this.prefix = prefix;

// this.suffix = suffix;

}

string name;

string prefix;

// string suffix;

string[] tryingSuffixes;

static bool compareSuffToPref(string suff, string pref) {

return suff.Equals(pref);

}

}