Министерство образования и науки Российской Федерации

Федеральное государственное автономное образовательное учреждение высшего образования

**«Уральский федеральный университет**

**имени первого Президента России Б.Н. Ельцина»**

**Институт математики и компьютерных наук**

**Кафедра алгебры и дискретной математики**

**РАЗЛИЧЕНИЕ СЛУЧАЙНЫХ СЛОВ МАЛЕНЬКИМИ КОНЕЧНЫМИ АВТОМАТАМИ**

Допустить к защите:

Зав. кафедрой

доктор физико-математических наук,

профессор

Волков М.В.

Выпускная квалификационная работа

студента 4 курса

Карповой О.Д.

Научный руководитель:

доктор физико-математических наук,

профессор

Шур А.М.

Екатеринбург

2016

**СОДЕРЖАНИЕ**

[ВВЕДЕНИЕ 3](#_Toc453423445)

[Глава 1. ИСПОЛЬЗУЕМЫЕ ФАКТЫ 5](#_Toc453423446)

[1.1 Длина рассматриваемых строк 5](#_Toc453423447)

[1.2 Независимость от размера алфавита 6](#_Toc453423448)

[1.3 Одинаковое количество нулей и единиц 6](#_Toc453423449)

[1.4 Счетчик подстрок 6](#_Toc453423450)

[1.5 Случайные строки 6](#_Toc453423451)

[1.6 Различие вблизи начала строки 6](#_Toc453423452)

[1.7 Различие вблизи конца строки 7](#_Toc453423453)

[1.8 Расстояние Хэмминга 8](#_Toc453423454)

[Глава 2. ПОСТАВЛЕННЫЕ ЭКСПЕРИМЕНТЫ 9](#_Toc453423455)

[2.1 Рассматриваемые автоматы и строки 9](#_Toc453423456)

[2.2 Фильтрация строк случайными автоматами 10](#_Toc453423457)

[2.3 Перестановочные автоматы 11](#_Toc453423458)

[2.3.1 Эксперимент на основе графа Рози 12](#_Toc453423459)

[2.3.2 Ограничение на структуру строки 15](#_Toc453423460)

[2.4 Произвольные автоматы 17](#_Toc453423461)

[2.4.1 Тождества для автоматов с четырьмя состояниями 18](#_Toc453423462)

[2.4.2 Тождества для автоматов с пятью состояниями 18](#_Toc453423463)

[ЗАКЛЮЧЕНИЕ 21](#_Toc453423464)

[СПИСОК ЛИТЕРАТУРЫ 23](#_Toc453423465)

[ПРИЛОЖЕНИЯ 24](#_Toc453423466)

# **ВВЕДЕНИЕ**

Пусть *А* = (Σ, Q, δ, s, T) – детерминированный конечный автомат, где

Σ – входной алфавит, из которого формируются слова, принимаемые автоматом, Σ ≠ Ø,

Q – множество состояний автомата, Q ≠ Ø,

δ – функция переходов, определенная как отображение δ: Q × Σ → Q,

s ­– начальное состояние, s ∈ Q

T – множество терминальных (конечных) состояний, T ⊆ Q.

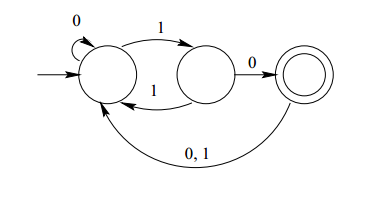
**Определение.** Автомат *А* – перестановочный, если переход из любого состояния по любому символу является перестановкой состояний, или, что тоже самое, для любого символа x из Q и любых состояний q и p δ(q, x) ≠ δ(p, x).

**Определение.** Автомат принимает слово, если по окончании его обработки он находится в терминальном состоянии.

**Определение.** Пусть u и v – слова над алфавитом Σ. Говорят, что автомат различает слова u и v, если он принимает одно из них и не принимает другое.

Пару слов (u, v) будем называть тождеством для некоторого автомата, если он либо принимает и u, и v, либо отвергает и u, и v, то есть не различает их.

Обозначим за sep(u, v) количество состояний в минимальном детерминированном конечном автомате, различающем u и v, за sep\_p(u, v) – количество состояний в перестановочном автомате, различающем u и v.

Например, автомат на рисунке 1 различает слова 0010 и 1000.

*Рисунок 1.*

Однако, можно проверить, что ни один автомат из двух состояний различить эти слова не сможет. Значит, sep(0010, 1000) = 3.

Очевидно, что sep(u, v) = sep(v, u), и sep\_p(u,v) = sep\_p(v, u).

Обозначим

S(n) = ,

и

S\_p(n) = .

Задача о различении слов, известная как Separating Words Problem, состоит в том, чтобы найти хорошую асимптотическую оценку функции S(n).

Другими словами, сколько состояний должно быть в автомате, различающем две строки длины n?

Эту классическую задачу в 1986 году сформулировали Павел Горальчик и Вацлав Коубек [1,2]. Они же доказали, что S(n) = o(n). Позже этой задачей занимался Джон Робсон [3,4]. В 1989 году он доказал, что

S(n) = O(n2/5(log n)3/5) для произвольных автоматов, и в 1996 году опубликовал статью, в которой показал, что S\_p(n) = O(n1/2). Однако зазор между верхней и нижней границей до сих пор остается открытой задачей.

Исследования данной задачи сводятся к поиску тождеств для автоматов.

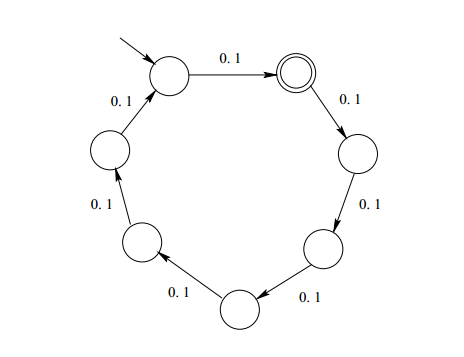
Цель моих экспериментов – наблюдение за тем, как различают слова автоматы из четырех – шести состояний, и поиск тождеств для них среди строк длины 40 – 60. Успешные эксперименты помогут понять, как ведет себя функция S(n) при малых аргументах.

# **Глава 1. ИСПОЛЬЗУЕМЫЕ ФАКТЫ**

## **1.1 Длина рассматриваемых строк**

Рассматриваемые мной строки одинаковой длины. Этот случай более интересен, поскольку из классической теоремы Чебышёва о простых числах следует, что для слов u и v разной длины всегда существует маленькое простое число p (р = O(log n), где n – минимум из длин u и v), такое, что длины u и v различны по модулю р. Чтобы различить строки в данном случае, можно использовать автомат, считающий длину своего входа по модулю р. Другими словами, sep(u, v) = O(log n), и существует бесконечное количество случаев, когда sep(u, v) = Ω(log n).

Например, предположим, что |u| = 22, |v| = 52. Тогда |u| ≡ 1 (mod 7), |v| ≡ 3 (mod 7). Так что мы можем различить u и v, автоматом, приведенным на рисунке 2.



*Рисунок 2.*

В дальнейшем, я буду рассматривать пары строк одинаковой длины. И в связи с этим, можно переопределить

S(n) = ,

и

S\_p(n) = .

## **1.2 Независимость от размера алфавита**

Без ограничения общности можно рассматривать строки над двухбуквенным алфавитом. Для строк над алфавитом большего размера, различных для некоторого автомата, существует гомоморфизм, сопоставляющий им строки той же длины над двухбуквенным алфавитом, различающиеся автоматом с тем же количеством состояний. Любой автомат, различающий двоичные строки, можно преобразовать в автомат, различающий строки над исходным алфавитом, не меняя количество его состояний. Поэтому в дальнейшем я буду рассматривать строки над алфавитом {0, 1}.

## **1.3 Одинаковое количество нулей и единиц**

**Определение.** Весом Хэмминга называют количество ненулевых символов в строке.

Если две строки имеют различное количество нулей и единиц, они могут быть различены автоматом, считающим вес Хэмминга у входной строки по модулю р, где р – небольшое простое число, р = O(log n).

## **1.4 Счетчик подстрок**

Если некая подстрока длины k встречается в строках разное количество раз, то эти строки можно различить автоматом с количеством состояний

O(k log n).

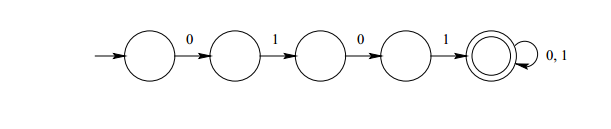
## **1.5 Случайные строки**

Для двух случайно выбранных строк u и v sep(u, v) = O(1).

## **1.6 Различие вблизи начала строки**

Если слова u и v различаются в позиции d, считая с начала, то

sep(u, v) ≤ d + 2.

Например, чтобы различить слова 0101101010101010 и 0100101010101010 можно построить автомат, изображенный на рисунке 3.

*Рисунок 3.*

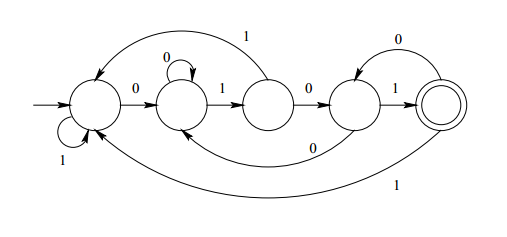
На данном рисунке опущены переходы в «мертвое» состояние.

Стоит заметить, что автомат, различающий слова, отличающиеся близко к началу, не является перестановочным. Это утверждение нам понадобится чуть позже.

## **1.7 Различие вблизи конца строки**

Если наибольший общий суффикс u и v имеет длину k, то u и v можно различить, используя автомат с k+2 состояниями. Отсюда вытекает следующее.Если слова u и v различаются в позиции d, считая с конца, то sep(u, v) ≤ d + 1.

К примеру, чтобы отличить 11111010011001010101 от 11111011010010101101, мы можем построить автомат, принимающий слова, которые заканчиваются на 0101, как показано на рисунке 4.



*Рисунок 4.*

Автомат, различающий строки, отличающиеся близко к концу, также не является перестановочным.

## **1.8 Расстояние Хэмминга**

**Определение.** Пусть u и v – строки одинаковой длины. u и v имеют расстояние Хэмминга d, если они различаются в d позициях и совпадают в остальных.

Если две строки имеют расстояние Хэмминга d, то существует простое число p = O(d log n) и позиция i, в которой строки отличаются, и которая по модулю p не равна никакой другой такой позиции. Путем вычисления четности входных символов в позициях, сравнимых с i по модулю p, можно различить строки, используя автомат с O(d log n) состояниями.

# **Глава 2. ПОСТАВЛЕННЫЕ ЭКСПЕРИМЕНТЫ**

## **2.1 Рассматриваемые автоматы и строки**

В своих экспериментах я буду представлять автомат не как пятерку, а как четверку составляющих, опуская множество терминальных состояний.

В связи с этим, будем говорить, что автомат *А* = (Σ, Q, δ, s) различает слова u и v, если он заканчивает их чтение в разных состояниях, то есть s.u ≠ s.v, где s.u – состояние, в которое приходит автомат, читая слово u из состояния s. Это эквивалентно первому определению о различении слов автоматом, поскольку если s.u ≠ s.v, то можно построить такой автомат *В* = (Σ, Q, δ, s, T), где s.u ∈ T, а s.v ∉ Т. Опустить множество терминальных состояний удобно для перебора автоматов, это значительно сокращает количество автоматов.

Также, имеет смысл рассматривать только автоматы, все состояния которых достижимы, поскольку если в автомате есть недостижимые состояния, его можно рассматривать как автомат с меньшим количеством состояний.

**Алгоритм 1.** *Проверка состояний автомата на достижимость из начального состояния.*

*Вход:* автомат A, состояние q ∈ Q.

*Выход:* достижимы ли из начального состояния автомата А остальные вершины.

used ← {q}

for v ∉ used, v ∈ {δ(q,0), δ(q, 1)}

reach(v)

Запуск:

A ← (Σ, Q, δ, s, T)

reach(s, δ)

return (used = Q)

**Наблюдение.** Если пара (u, v) является тождеством для всех автоматов из N состояний, она будет тождеством для всех автоматов из M < N состояний.

Напомню также, что я рассматриваю строки одинаковой длины над алфавитом {0, 1}, количества нулей (и единиц соответственно) в строках совпадают (см. п.1.2 и п.1.3).

## **2.2 Фильтрация строк случайными автоматами**

При проведении экспериментов может возникнуть ситуация, когда пар-кандидатов на проверку на тождество очень много. Если у автомата совсем маленькое количество состояний (скажем, 2 или 3), справиться за разумное время, конечно, можно. Но уже для автоматов из 4 и более состояний процесс может затянуться.

Однако, список кандидатов можно сократить. Для этого можно каждую пару-кандидат попытаться различить небольшим количеством случайных автоматов, получив при этом список «трудноразличимых» пар. Как показала практика, полученный список действительно меньше исходного, а значит и его обработка пройдет быстрее.

**Алгоритм 2.** *Фильтрация пар-кандидатов на тождества случайными автоматами.*

*Вход:* pairs – список пар-кандидатов на тождества, n – количество состояний в автомате, getRandomAutomata – функция, возвращающая случайный автомат с n состояниями, filtersNumber – число случайных автоматов-фильтров.

*Выход:* отфильтрованный список пар.

filteredList ← Ø

foreach pair ∈ pairs

separated ← false

for i=0 to filtersNumber-1

automata = getRandomAutomata(n)

if(automata.Separates(pair)

separated ← true

конец цикла

if(not separated)

filteredList = filteredList ∪ {pair}

return filteredList

К примеру, на рисунке 5 изображен график зависимости времени различения от количества пар строк.

*Рисунок 5.*

*График зависимости времени различение от количества пар слов.*

Верхняя линия отвечает за различение без использования фильтрации, нижняя – за различение с предварительной фильтрацией. Различение проводилось произвольными автоматами с пятью состояниями. Строки-кандидаты – тождества для произвольных автоматов с четырьмя состояниями. Количество случайных автоматов-фильтров – 20.

## **2.3 Перестановочные автоматы**

Чтобы сократить количество перебираемых автоматов, большую часть экспериментов я посвятила различению слов перестановочными автоматами с маленьким количеством состояний и поиску тождеств для них.

Так как автоматы, которые различают строки, отличающиеся в позиции, близкой к началу или концу, не являются перестановочными, строки, имеющие общий префикс или суффикс, в данном контексте не интересны. Поэтому для перестановочных автоматов имеет смысл рассматривать те пары, где строки начинаются с разных символов, а также заканчиваются разными символами.

Для перебора перестановочных автоматов использовался список неизоморфных перестановочных автоматов с пятью состояниями, расположенный в приложении 1.

### **2.3.1 Эксперимент на основе графа Рози**

Известно, что автоматы-счетчики подслов – это перестановочные автоматы с kp состояниями, где k - длина подслова и p - модуль, по которому количества подслов не равны. Таким образом, первой идеей было рассмотреть пары строк с одинаковым набором подстрок длины не больше 5. Для этого я использовала граф Рози пятого порядка.

**Определение.** Граф Рози порядка k – это граф, вершинами которого являются подстроки длины k, а вершины u и v соединены ребром тогда и только тогда, когда u[2, k] = v[1, k – 1], где u[i, j] – подстрока строки u, начинающаяся в позиции i и заканчивающаяся в позиции j.

**Алгоритм 3.** *Построение графа Рози для строки.*

*Вход:* строка u, k – порядок графа Рози.

*Выход:* граф Рози, соответствующий данной строке.

n ← длина u

prev ← u[1, k]

E ← Ø, V ← {prev}

for i = 1 to n – k

V ← V ∪ {u[i, i+k]}

E ← E ∪ {(prev, u[i, i+k])}

prev ← u[i, i+k]

**Идея эксперимента.** Для случайно выбранной строки длины N (N = 40 ÷ 50 в проведенных мною экспериментах) строится соответствующий ей граф Рози порядка k. Затем в нем я нахожу два цикла с общей вершиной и меняю их местами. Таким образом получается пара выбранной случайно строке. Такие пары я пробую отфильтровать способом, описанным в первом разделе, десятью случайными перестановочными автоматами из пяти состояний.

**Алгоритм 4.** *Функция перестановки циклов в маршруте графа Рози.*

*Вход:* маршрут p в графе Рози, вершина q.

*Выход:* маршрут, в котором поменяны местами два цикла с общей вершиной v.

(i1, i2, i3) ← номера позиций вершины v в маршруте.

q ← p[1, i1) + p[i2, i3) + p[i1, i2) + p[i3, n], где

p[i, j] – отрезок маршрута с i по j позицию, p[i, j) – конец не включается.

return q

**Результат.** Для данного эксперимента была использована функция фильтрации пар-кандидатов на тождества, описанная в п.2.2. Количество пар-кандидатов 100 000. Порядок графа Рози, а также длины строк выбирались различные. С результатами и зависимостями от порядка графа и длины строки можно ознакомиться на рисунках 6 и 7.

В ходе такого эксперимента тождеств для перестановочных автоматов с пятью состояниями мне найти не удалось. Возможно, это связано с тем, что получаемые мной строки имели маленькое расстояние Хэмминга. Известно, что если два слова u и v имеют расстояние Хэмминга меньше d, то sep(u, v) ≤ O(d log n), где n – длина рассматриваемых слов.

*Рисунок 6.*

*Зависимость количества пар трудноразличимых слов от длины строки. Порядок графа Рози 4.*

*Рисунок 7.*

*Зависимость количества пар трудноразличимых слов от порядка*

*графа Рози. Длина строки 40.*

### **2.3.2 Ограничение на структуру строки**

В поиске других идей я обратилась к уже известным тождествам для перестановочных автоматов из 5 и 6 состояний, найденных в 2013 Андреем Рощенюком [5]. Тождества были найдены среди пар, где второй строкой бралась обратная запись первой строки. Обратную запись строки u в дальнейшем буду обозначать как reverse(u).

Известное тождество для перестановочных автоматов с пятью состояниями:

u = xy(xyyx)3(yxxy)2yx(yxxy)2, v = reverse(u).

Известное тождество для перестановочных автоматов с шестью состояниями:

u = (xy)6(yx)10(xy)4, v = reverse(u).

Здесь и далее x, y ∈ {0, 1}, x ≠ y.

**Идея эксперимента.** Я заметила, что строка из пары состоит из двух блоков четной длины, и внутри каждого такого блока количество 0 и 1 совпадает. Более того, в них не содержится подстрок 000 и 111. Это натолкнуло меня на мысль выбирать случайную строку следующим образом: выбирать первую и вторую половины строк среди строк длины N/2, не содержащих подстроки 000 и 111, с равным количеством 0 и 1. При этом при склейке строк также проверять, чтобы посередине не возникло больше двух нулей или единиц подряд.

**Алгоритм 5.** *Построение слов с равным количеством нулей и единиц, не содержащих подстроки 000 и 111.*

*Вход:* n – длина строки.

*Выход:* список слов длины n с равным количеством нулей и единиц, не содержащих 000 и 111.

list ← Ø

str ← startString(n)

while(str.Length == n)

if(str.HasNo111And000 and str.HasEqualCountOf0And1)

list ← list ∪ {str}

str ← str.GetNext()

return list

**Алгоритм 6.** *Построение случайной строки, не содержащей подстрок 000 и 111, где каждая ее половина содержит одинаковое количество нулей и единиц.*

*Вход:* n – длина искомой строки.

*Выход:* строка, не содержащая подстрок 000 и 111, и каждая ее половина содержит одинаковое количество нулей и единиц.

halfs ← GetWordsWithout000and111WithEqualCount0And1(n/2)

str ← “”

while(not str.HasNo111And000)

str ← halfs[rand1] + halfs[rand2]

return str

Ограничив таким образом выбор случайной строки, и принимая в пару обратную запись строки, я получила более существенный результат.

В данном эксперименте использовались перестановочные автоматы с пятью состояниями, однако полученные тождества я проверяла также перестановочными автоматами с шестью и семью состояниями.

В пару строке – ее обратная запись.

**Результат.** Для строк длины 40 оказалось возможным перебрать все подобные пары за разумное время. Так, я получила 10 тождеств, которые, однако, оказались модификациями уже известных. Под модификацией здесь я предполагаю замену символа на обратный, а также добавление общих суффикса и префикса.

После я взялась за строки длины 48. Перебрать их все оказалось уже гораздо дольше, поэтому я ограничилась выбором случайных строк. После серии подобных экспериментов мне удалось найти тождества для перестановочных автоматов с пятью и шестью состояниями, которые нельзя получить модификацией уже известных:

Для перестановочных автоматов с пятью состояниями:

(xyyx)2(yxxy)5(xyyx)3(yxxy)

(xyyx)6(yxxy)5

(xyyx)2(yxxy)2(yx)2(yxxy)3(xyyx)2(xy)2

(xyyx)2(xy)2(yx)2(xyyx)2(yxxy)6

(xy)3(yx)3(xy)3(yx)3(xy)3(yx)4(xyyx)2(xy)

(xyyx)5(yx)(yxxy)2(xy)(xyyx)2(yxxy)3

(xyyx)2(xy)(xyyx)6(yxxy)3(yx)2(xy)

Для перестановочных автоматов с шестью состояниями:

(xy)7(yx)12(xy)5

(xy)10(yx)12(xy)2

(xy)4(yx)5(xy)6(yxxy)(yx)6(xy)

В данных списках дано только одно слово из пары, второе получается разворотом первого.

## **2.4 Произвольные автоматы**

Автоматы, которые различают слова, отличающиеся в самом начале или в самом конце, не являются перестановочными. Поэтому все тождества для перестановочных автоматов – пары строк, начинающихся и заканчивающихся по-разному. Соответственно, чтобы превратить тождество для перестановочных автоматов в тождество для произвольных автоматов, нужно дописать к строкам одинаковые префиксы и суффиксы. Вспомним тот факт, что если строки различаются в позиции d, считая с начала, то они различимы автоматом, для которого |Q| ≤ d+2, а различающиеся в позиции d с конца – автоматом, для которого |Q| ≤ d+1(см. в разделе используемые факты). Значит, чтобы исключить различение автоматами из k состояний около начала и конца строк, префикс должен иметь длину не менее k-1, а суффикс - длину не менее k.

**Идея эксперимента.** Допишем в начало и конец пары-тождества для перестановочных автоматов одинаковые префиксы и одинаковые суффиксы. Суффикс (равно как и префикс) состоит из одинакового количества нулей и единиц.

Для автоматов с четырьмя и пятью состояниями эксперименты получились немного разными.

## **2.4.1 Поиск тождеств для автоматов с четырьмя состояниями**

**Идея эксперимента.** К тождествам для перестановочных автоматов с пятью состояниями дописать одинаковые префикс и суффикс и пытаться различить их всеми автоматами с четырьмя состояниями. Суффикс берется как обратная запись префикса.

**Результат.** В ходе такого эксперимента удалось получить несколько сотен тождеств для произвольных автоматов из четырех состояний. Длины получившихся строк, однако, вышли не менее 52.

Тождество длины 52 получилось следующим:

u = (xy)9(yx)10(xy)4(yx)3, v = reverse(u)

Преимущество такого тождества в том, что оно является тождеством для перестановочных автоматов с пятью состояниями, а значит есть шанс продолжить его до тождества для произвольных автоматов с пятью состояниями.

Кратчайшее тождество такого вида (суффикс равен обратной записи префикса) для произвольных автоматов с четырьмя состояниями мне удалось найти следующее:

u = (xyy)(yxx)2(y)9(xxy)2(yy)(xxy)2(yyx), v = reverse(u).

Слово здесь имеет длину 35. Данное тождество было получено из кратчайшего тождества для перестановочных автоматов с четырьмя состояниями (u = 00000011011, v = 11011000000). Такое тождество не является тождеством для перестановочных автоматов с пятью состояниями.

## **2.4.2 Поиск тождеств для автоматов с пятью состояниями**

**Идея эксперимента.** Для найденных в первом эксперименте тождеств для произвольных автоматов с 4 состояниями провести следующий процесс.

На каждой итерации, перебирая все автоматы с 5 состояниями по порядку, находим первый автомат, который различает пару. Затем ищем такие суффикс и префикс для строк, чтобы при добавлении их к строкам, этот автомат перестал их различать. Далее проверяем, не стал ли различать получившуюся строку какой-либо из предыдущих автоматов и затем ищем уже следующий автомат, который их различит.

**Алгоритм 7.** *Получение тождества для произвольных автоматов с k состояниями из пары-тождества для перестановочных автоматов с k состояниями путем жадного дописывания общих суффикса и префикса.*

*Вход:* k – число состояний в автомате, (u, v) – пара-тождество для перестановочных автоматов с k состояниями, t – ограничение для длины суффикса на одной итерации.

*Выход:* пара-тождество для произвольных автоматов с k состояниями.

while(true)

hardAutomata ← GetFirstAutomataSeparated(u, v)

prefix ← getFirstPrefix(k+1)

suffix ← getFirstSuffix(k)

goodSuff ← Ø

goodPref ← Ø

while(|goodSuff| < t)

u1 ← suffix + u + prefix

v1 ← suffix + v + prefix

if( not hardAutomata.Separates(u1, v1))

goodPref ← goodPref ∪ {prefix}

goodSuff ← goodSuff ∪ {suffix}

suffix ← GetNextSuffix(suffix)

prefix ← GetNextPrefix(prefix)

maxSeparatedNumber ← max i {GetFirstAutomataSeparated(u1, v1) |

u1 ← goodPref[i] + u + goodSuff[i],

v1 ← goodPref[i] + v + goodSuff[i]}

i\* = arg(maxSeparatedNumber)

if(maxSeparatedNumber > номер последнего автомата)

Успех! Конец. (u1, v1) – искомое тождество.

if(maxSeparatedNumber > hardAutomata.Number)

u ← goodPref[i\*] + u + goodSuff[i\*]

v ← goodPref[i\*] + v + goodSuff[i\*]

Рассматривались разные варианты взятия «следующей строки»: следующая в лексикографическом порядке с ограничением на равенство 0 и 1 и без него. Суффикс также брался по-разному: при одних запусках как независимая от префикса строка, при других – обратная запись префикса, при третьих – равная префиксу строка. Успеха я смогла добиться, когда рассматривала префиксы с равным количеством нулей и единиц и суффиксы, равные обратной записи префикса.

**Результат.** Таким жадным алгоритмом я нашла тождество для произвольных автоматов с 5 состояниями длины 128:

(xyx)3(yx)(xy)3(yx)2(xxy)(yx)2(xy)4(yx)2(xy)2\_(xy)6(yx)10(xy)4\_(yx)2(xy)2(yx)4

(xy)2(yxx)(xy)2(yx)3(xy)(xyx)3

И длины 124:

(xx)(yx)2(xyyy)(yx)2(y)(yx)2(xxy)(yx)2(xy)4(yx)2(xy)2\_(xy)6(yx)10(xy)4\_(yx)2(xy)2

(yx)4(xy)2(yxx)(xy)2(y)(xy)2(yyyx)(xy)2(xx)

К сожалению, этот алгоритм достигает успеха не на каждой строке. Во многих случаях при дописывании префикса и суффикса, новую пару строк начинает различать автомат, который прежде этого не делал. Дело в следующем. Пусть автомат А = (Σ, Q, δ, q0) не отличил u от v. Допишем префикс х к u и v. Может возникнуть ситуация, что в конце чтения префикса х автомат окажется в состоянии q1, а автомат B = (Σ, Q, δ, q1) уже умеет отличать u от v.

# **ЗАКЛЮЧЕНИЕ**

В ходе проведенных экспериментов мне удалось найти некоторое количество тождеств для перестановочных и произвольных автоматов с четырьмя-шестью состояниями, а также разработать идеи поиска таких тождеств.

Есть разные направления продолжения моих экспериментов, и вот несколько из них:

1. Оптимизация использованных алгоритмов и сокращение перебора.

Это очень важно, когда дело касается перебора длинных строк, а также перебора автоматов. К примеру, перебор только неизоморфных автоматов с пятью состояниями существенно сократил длительность некоторых экспериментов. А значит для перебора автоматов с большим количеством состояний такую оптимизацию будет очень выгодно провести.

1. Расширение экспериментов с графом Рози.

Проведенный мною эксперимент с обменом циклов достаточно узок. Можно его расширить, пытаясь, например, поменять циклы в нескольких разных вершинах графа, или преобразовать строку после смены цикла каким-либо другим образом, или же найти другой путь в графе на том же наборе вершин, такой, что каждая вершина будет встречаться в этом пути ровно столько раз, сколько встречалась в исходном.

Я считаю, что в этом направлении можно найти что-то интересное для различения автоматами слов.

1. Есть много различных вариантов запуска жадного алгоритма, а именно

вариантов выбора общих суффиксов и префиксов. А также осталось много пар строк, к которым данный алгоритм не применялся. Кто знает, может одна из таких строк даст более интересный вариант.

1. Ускорить работу над бинарными строками может помочь их

представление в виде чисел, если считать строку записью числа в двоичной системе счисления. Тогда некоторые операции над строками можно будет заменить операциями над числами, что может ускорить процесс обработки.

Вопрос об асимптотике функции S(n), равно как и вопрос о длинах кратчайших тождеств, остается открытым до сих пор. Конкретных тождеств найдено не очень много, закономерность в них также до сих пор не известна. А значит пространство для экспериментов в данной области еще есть.

# **СПИСОК ЛИТЕРАТУРЫ**

[1] P. Goralčík and V. Koubek. On discerning words by automata // In L. Kott, editor, Proc. 13th Int’l Conf. on Automata, Languages, and Programming (ICALP), volume 226 of Lecture Notes in Computer Science, pages 116–122. Springer-Verlag, 1986.

[2] E.D. Demaine, S. Eisenstat, J. Shallit, D.A. Wilson. “Remarks on Separating Words // Descriptional Complexity of Formal Systems (DCFS 2011), P. 147-157. LNCS Vol. 6808. Springer, 2011.

[3] J.M. Robson. Separating words with machines and groups// RAIRO Inform. Théor. App. 30, 81–86 (1996)

[4] J. M. Robson. Separating strings with small automata // Inform. Process. Lett.,

30:209–214, 1989.

[5] А.Рощенюк, Поиск тождеств от двух переменных в группах перестановок. Курс. раб., ИМКН УрФУ, 2013.

# **Приложение А**

Пусть в перестановочном автомате с множеством вершин {1,2,3,4,5} X - это 0-перестановка, Y - 1-перестановка.

Автоматы, в которых X коммутирует с Y, рассматривать не надо: они ничего не разделят, поскольку нулей и единиц в рассматриваемых словах поровну.

Перестановки обозначаем произведениями циклов, типа (124)(35).

По списку длин циклов перестановки делятся на 6 классов : 2, 22, 3, 23, 4, 5

Для каждой пары классов берем X из первого, Y из второго и смотрим неизоморфные способы взаимного расположения.

1. X из 2; б.о.о. X=(12).

Y из 2: Y=(23) (остальные случаи изоморфны этому или Y коммутирует с X; такая оговорка верна и для всех последующих случаев)

Y из 22: Y=(23)(45) или (13)(24)

Y из 3: Y=(123) или (134)

Y из 23: Y=(123)(45) или (134)(25)

Y из 4: Y=(1234) или (1324) или (2345)

Y из 5: Y=(12345) или (13245)

2. X из 22; б.о.о. X=(12)(34)

Y из 2: см. выше (в п.1 поменять роли X и Y)

Y из 22: Y=(13)(24) или (12)(35) или (13)(25)

Y из 3: Y=(123) или (125) или (135)

Y из 23: Y=(123)(45) или (135)(24) или (125)(34)

Y из 4: Y=(1234) или (1235) или (1325) [не (1324), потому что (1324)(1324)=(12)(34)]

Y из 5: Y=(12345) или (13245) или (13425)

3. X из 3; б.о.о. X=(123)

Y из 2, 22: см. выше

Y из 3: Y=(145) или (124) или (214)

Y из 23: Y=(145)(23) или (124)(35) или (214)(35)

Y из 4: Y=(1234) или (3214) или (1245) или (2145) или (1425)

Y из 5: Y=(12345) или (32145) или (12435) или (21435)

4. X из 23; б.о.о. X=(123)(45)

Y из 2, 22, 3: см. выше

Y из 23: Y=(145)(23) или (124)(35) или (214)(35)

Y из 4: Y=(1234) или (3214) или (1245) или (2145) или (1425)

Y из 5: Y=(12345) или (32145) или (12435) или (21435)

5. X из 4; б.о.о. X=(1234)

Y из 2, 22, 3, 23: см. выше

Y из 4: Y=(1243) или (1235) или (2135) или (1325) или (2315) или (3215) или (3125) [(1324) дает вариант, изоморфный (1243)]

Y из 5: Y=(12345) или (43215) или (12435) или (12543) или (12453) или (21354)

6. X из 4; б.о.о. X=(1234)

Y из 2, 22, 3, 23, 4: см. выше

Y из 5: Y=(12354) или (12453) или (12543) или (13254)

Всего (с учетом удвоения за перемену ролей X и Y, когда они из разных классов) - 121 автомат.

Для проверки разделимости у каждого автомата нужно каждую вершину рассматривать как начальную.