О задачах извлечения корня из заданного конечного языка

Б. Ф. Мельников, А. А. Мельникова

Аннотация—**В статье рассматриваются только конечные языки.**

На основе стандартного определения произведения (конкатенации) языков вводится целая неотрицательная степень языка. Извлечение корня – обратная к ней операция, причём её можно определить несколькими разными способами. Несмотря на простоту формулировки задачи, авторы не смогли найти в литературе (а также в Интернете) какого-либо её описания, в том числе даже её постановки. Но, несмотря на это, мы считаем, что весь материал статьи (в том числе доказанная теорема) о возможном изменении корня-ответа является очень лёгким, и вызывает недоумение тот факт, что в известных авторам монографиях нет хотя бы формулировок подобных задач.

Бóльшая часть материала настоящей статьи посвящена самому простому варианту формулировки – корню 2-й степени для 1-буквенного алфавита – но при этом многие положения статьи обобщаются и на более сложные случаи. По-видимому, для возможного в будущем описания полиномиального алгоритма решения хотя бы одной из описанных постановок задач извлечения корня сначала нужно действительно подробно разобрать такой частный случай, то есть: либо описать необходимый полиномиальный алгоритм, либо, наоборот, показать, что задача принадлежит классу NP-полных задач.

Итак, в настоящей статье мы не предлагаем полиномиального алгоритма для рассматриваемых задач – однако модели, описываемые здесь, должны помочь в построении соответствующих эвристических алгоритмов их решения. Подробное описание возможного дальнейшего применения подобных эвристических алгоритмов выходит за рамки настоящей статьи – но несколько аргументов о таком возможном применении можно провести уже сейчас. Во-первых, достаточно очевидна возможность применения таких алгоритмов в задачах криптографии и криптоанализа. Во-вторых, даже если для рассматриваемой задачи извлечения корня имеется полиномиальный алгоритм – то он, по-видимому, довольно сложен, и должен быть назван «труднорешаемой задачей». В-третьих, рассматриваемая задача несложно сводится к NP-полной задачи из области теории графов – задаче поиска клики; при этом обратное сведéние не доказано; также упомянем задачу о покрытии множества.

Ключевые слова—**формальные языки, итерации языков, извлечение корня, алгоритмы, булеан.**

I. ВВЕДЕНИЕ

Определение произведения двух языков вводится стандартным образом ([1], одно из альтернативных названий этой операции – конкатенация). На основе введённого определения произведения – также стандартным алгебраическим методом – вводится (целая неотрицательная) степень языка. Извлечение корня – обратная операция, причём её можно определять разными способами, некоторые подробности см. далее. В статье мы будем рассматривать задачу извлечения корня из заданного конечного языка.

Несмотря на простоту формулировки задачи, авторы не смогли найти в литературе (а также в Интернете) какого-либо её описания – в том числе *даже её постановки*. Конкретнее, при поиске в Интернете – причём и на английском, и на русском языках – несколько поисковиков выдают такие результаты.

- Либо статьи, в которых слово «корень» используется для корня какой-то заданной функции (равенства её нулю), чаще всего – для корня заданного многочлена.
- В качестве частного случая предыдущего пункта (либо в качестве совершенно иной задачи) можно рассматривать поиск корня из какого-то заданного вещественного числа.

Во всех случаях в обоих упомянутых пунктах требуется вычисление таких значений с заданной точностью. См. [2] и мн. др. Конечно, это тоже очень интересные задачи, для них описываются, среди прочих, нетривиальные параллельные алгоритмы – однако такие задачи совершенно не связаны с нашей тематикой.

Продолжим описание найденных в Интернете результатов.

- Упомянем статьи по тематике, связанной с «естественными» языками, в которых слово «корень» воспринимается не в алгебраическом смысле – а в лингвистическом [1].
- Также поисковки могут находить наши статьи – [3], [4] и др. – в которых только начато решение задачи извлечения корня из языка, а также решаются связанные с нею задачи.

При этом мы считаем, что весь материал настоящей статьи (в том числе доказанная ниже теорема) является *очень лёгким* – и вызывает большое недоумение тот факт, что хотя бы формулировок подобных задач нет в известных авторам монографиях [1], [5], [6], [7], [8] и др.

Бóльшая часть материала настоящей статьи посвящена самому простому варианту формулировки – корню 2-й степени для 1-буквенного алфавита – но при этом многие положения статьи обобщаются и на более сложные случаи. И, по-видимому, для возможного в будущем описания полиномиального алгоритма решения хотя бы одной из описанных постановок задач сначала нужно в первую очередь подробно разобрать такой частный

Статья получена 23 января 2023 г.

Борис Феликсович Мельников, Университет МГУ – ППИ в Шэньчжэне (bormel@smbu.edu.cn).

Александра Александровна Мельникова, Димитровградский инженерно-технологический институт – филиал Национального исследовательского ядерного университета «МИФИ» (super-avahi@yandex.ru).

[1] “An effective multilingual stemmer based on the extraction of the root” и т. п. Ссылок на конкретные статьи, конечно, приводить не будем.

случай (в наших обычных обозначениях – найти $\sqrt[2]{A}$ при $|\Sigma| = 1$) – то есть:

- либо описать необходимый полиномиальный алгоритм,
- либо, наоборот, показать, что задача принадлежит классу NP-полных задач.

Итак, в настоящей статье мы *не предлагаем* полиномиального алгоритма для рассматриваемых задач – однако *модели*, описываемые здесь, должны помочь в построении соответствующих *эвристических алгоритмов* их решения. Подробное описание возможного дальнейшего применения подобных эвристических алгоритмов, конечно, выходит за рамки настоящей статьи – но несколько аргументов о таком возможном применении можно провести уже сейчас:

- достаточно очевидна возможность применения таких алгоритмов в задачах криптографии и криптоанализа;
- даже если для рассматриваемой задачи извлечения корня имеется полиномиальный алгоритм – то он, по-видимому, довольно сложен, и по терминологии классической монографии [9] должен быть назван «труднорешаемой задачей»;
- рассматриваемая задача несложно сводится к NP-полной задачи из области теории графов – задаче поиска клики (также см. [9]); при этом обратное сведе́ние не доказано (возможно оно и есть, но, в любом случае, задача поиска корня близка к задаче о клике); также надо упомянуть задачу о покрытии множества, [9], [10] и мн. др..

Приведём содержание статьи по разделам.

Раздел II – предварительные све́дения, в нём приведены применяемые обозначения и варианты постановок решаемых в статье задач. *В разделе III* рассмотрено несколько простых моделей, предназначенных для описания алгоритмов решения задачи также в простом случае: для 1-буквенного алфавита и квадратного корня. При этом одна из соответствующих математических моделей формулируется на языке теории графов.

Подходы к решению простого варианта задачи (1-буквенный алфавит и квадратный корень) рассматриваются и *в разделе IV*: в нём мы определяем т. н. потенциальные корни, а для них – т. н. табуированные пары и табуированные гиперплоскости. Вообще, весь материал статьи начиная с раздела IV можно охарактеризовать так: мы рассматриваем *дополнение* графа, введённого в конце раздела III. При этом *в разделе V* приведено обобщение задачи для алфавита произвольной (конечной) мощности: на примерах показано, что применение описанных ранее моделей возможно и для такого случая.

Материал *раздела VI* описывает ту часть предмета статьи, которая, по-видимому, является единственной нетривиальной; в нём рассматривается добавление новой координаты к уже имеющемуся решению. Именно на основе результатов доказываемой в разделе теоремы мы можем утверждать, что поиск (хотя бы одного) корня может производиться как поиск вершины гиперкуба (булеана), ближе всего находящейся к максимальной вершине и при этом не входящей ни в одну из табуированных плоскостей.

В разделе VII рассматриваются несколько нетривиальных примеров. *Раздел VIII* – заключение. В нём приведены возможные направления дальнейших работ по рассмотренной в статье тематике, и, в частности, краткое описание плана ближайшей статьи – в которой, среди прочего, будет описана попытка обобщения предыдущих моделей на случай корня произвольной степени.

II. ПРИМЕНЯЕМЫЕ ОБОЗНАЧЕНИЯ И ВАРИАНТЫ ПОСТАНОВКИ ЗАДАЧИ

Начнём раздел с описания стандартных обозначений. Смысл слова «умножение» для рассматриваемых объектов понятен: это просто результат произведения двух языков; в частности, рассмотрим такой пример для 1-буквенного алфавита: если

$$C = \left\{ a^i \,\middle|\, i \in I \right\} \quad \text{и} \quad D = \left\{ a^j \,\middle|\, j \in J \right\}$$

(I и J – некоторые конечные подмножества целых неотрицательных чисел), то

$$C \cdot D = \left\{ a^k \,\middle|\, (\exists\, i \in I,\, j \in J)\,(k = i + j) \right\}.$$

Слово «умножение» будем в дальнейшем применять без кавычек – в то время как «деление» всегда в кавычках: это нестандартная операция, и её мы далее будем использовать толко для 1-буквенного алфавита и 1-элементного множества-«делителя».

На основе произведения естественным образом определяется степень языка; далее переходим к описанию операции извлечения корня [2].

Всюду при описании задач и рассматриваемых к ним примеров мы будем придерживаться следующих соглашений – о терминах и о применяемых константах.

- Первое. Всюду далее решается задача поиска языка X, являющегося корнем уравнения
$$X^M = A \tag{1}$$
(где язык A и число $M \geqslant 2$ заранее заданы [3]). Как следует из материала введения, в большей части статьи будем решать уравнение для $M = 2$.
- При этом *потенциальный корень* – это некоторое слово $u \in \Sigma^*$, такое что $u^M \in A$. В отличие от них (т. е. слов) – множество (язык) X будем называть корнем (без прилагательного «потенциальный»), если выполнено условие (1).
Очевидно, что найти все потенциальные корни можно за полиномиальное (относительно размера исходных данных задачи) время.
- Второе соглашение: N – количество потенциальных корней, нумеруем их от 1 от N, здесь мы используем «обычную индексацию Паскаля».
- Третье соглашение (оно относится только к случаю 1-буквенного алфавита). Если, как в приведённом выше примечании, «задача формируется» возведением в степень M некоторого языка (пусть это язык B; считаем что при решении получаем язык X), то

[2] На основе приведённых описаний можно привести *строгие формальные* определения всех употребляемых понятий – однако не будем этого делать.

[3] При этом рассматриваемые далее примеры, как правило, будут конструироваться так, как если бы был задан не язык A, а язык-ответ, т. е. обычно в наших обозначениях B. Поэтому в примерах мы заранее знаем ответ – точнее, один из возможных языков-ответов.

представление этого языка (как множества, B или X) выполняем перечислением от 0 до некоторого заданного n; здесь мы используем «индексацию Си-подобных алгоритмических языков».

При этом $M \cdot n + 1$ будет размерностью *исходного* языка A (например, в случае 2-й степени – нужно каким-то способом определить разряды от 0-го до $2n$-го).

Для того, что выше названо «третьим соглашением», мы во всех случаях (т. е. как для A, так и для B / X) будем применять 4 варианта обозначения множества слов:

- «самый обычный»: например, $\{ a, ab, ba, bab \}$, или

$$\{ \varepsilon{=}a^0, a{=}a^1, aaa{=}a^3, aaaaaa{=}a^6 \}. \quad (2)$$

Остальные три варианта записи будут применяться как для подмножеств множества потенциальных корней[4], так и – в случае 1-буквенного алфавита – для нескольких элементов множества

$$\{ \varepsilon, a, aa, \ldots a^{K-1}, a^K \}$$

(при этом обычно будет либо $K = n$, либо $K = 2n$). Приведём примеры – для языка (2):

- [0 1 3 6] – множество используемых степеней записываем в квадратных скобках в порядке возрастания;
- 1001011 – двоичное число, в котором разряды предыдущего списка отмечены 1; разряды двоичного числа нумеруем начиная с 0 справа налево;
- 75 – число, принадлежащее $\mathbb{N}_0$, представляющее собой десятичную запись предыдущего пункта.

Как уже отмечалось, те же самые обозначения будем использовать, когда $K = N$, и при этом мы рассматриваем подмножества множества потенциальных корней; однако в этом случае разряды двоичного числа мы будем записывать слева направо и нумеровать начиная с 1. Например, пусть всего 5 элементов, а подмножество – $\{2, 5\}$:

- [2 5], иногда просто 2 5, даже иногда 25 – неоднозначности это не вызовет; пустое множество обозначается обычно, т. е. $\varnothing$;
- 01001; пустым множеством здесь является последовательность нулей;
- 9; пустое множество – 0.

Отметим, что что недоразумений (неоднозначности прочтения) не возникнет – ни для выбора варианта обозначения языка, ни для понятий корень / потенциальный корень. Из контекста всегда будет понятно, в обычном ли смысле мы употребляем слово «корень» (корень из другого языка), либо как потенциальный корень (слово).

Итак, в статье в нескольких варантах будут использованы подмножества – но при этом только подмножества множества потенциальных корней (напомним, что мы предположили, что число их равно N) будем обычно рассматривать как вершины N-мерного гиперкуба.

Теперь определим несколько связанных с подобным гиперкубом вспомогательных понятий.

- Определение максимальной вершины гиперкуба естественно – это

$$(1, 1, \ldots, 1, 1).$$

- Для дальнейшего – в случае рассмотрения задачи извлечения корня 2-й степени – часто будут использованы *пары* потенциальных корней; в частности, т. н. табуированные пары (о них подробности далее). Для примера – если:
  - мы принимаем «третье соглашение» (напомним, что в нём мы рассматриваем только случай 1-буквенного алфавита)
  - и при этом рассматриваем множество [0 1 2 3 6] (оно действительно будет подробно рассмотрено в примерах в дальнейшей части статьи),
  - а пара – $\{2, 6\}$ (конечно, порядок элементов пары несущественен),

  то возможны ещё и такие обозначения этой пары:
  - $\overbrace{2, 6}$ (это просто их по значениям); это то же самое, что и $\overbrace{6, 2}$;
  - $\overbrace{\#3, \#5}$ (это их по номерам, считаем номера начиная с 1).
- Для некоторой табуированной пары табуированная гиперплоскость – это такая N−2-мерная гиперплоскость N-мерного гиперкуба, для которой обе координаты элементов пары равны 1.
- Заменим, что максимальная вершина гиперкуба входит в пересечение любого количества любых табуированных гиперплоскостей.
- Для некоторой точки гиперкуба $(b_1 b_2 \ldots b_N)$ обозначение $(b_1 b_2 \ldots b_N)_{+k}$ означает вершину, полученной из предыдущей путём замены k-й координаты на 1.

Перейдём к уточнению непосредственных постановок решаемых задач. Как было отмечено в [4], рассматривая для некоторого заданного конечного языка задачу извлечения корня заданной степени[5], мы фактичекски имеем дело не с одной задачей, а с целой *группой задач*, – поскольку имеется несколько вариантов для требуемого ответа:

- найти *любой* (корень из языка);
- найти *все*;
- найти *минимальный* (по некоторой метрике)[6];
- и т. п.

По этому поводу приведём пример с несколькими возможными корнями: у языка

$$[0\ 1\ 2\ 3\ 4\ 5\ 6\ 7\ 8]$$

имеется не только очевидный корень [0 1 2 3 4], но и корень [0 1 3 4]; понятно, что конструировать подобные примеры совсем несложно – гораздо сложнее обратная операция, которой мы в настоящей статье и занимаемся.

Также несложно понять, что все эти варианты рассматриваемой задачи очень близки между собой – и поэтому мы далее:

- либо не будем уточнять конкретную задачу,

[4] Для зафиксированного множества этих корней, мы ранее договорились, что их число равно N. Будем считать, что также зафиксирована и их нумерация.

[5] Либо максимально возможной степени. Разницы практически нет: в конкретном частном случае задачи максимально возможная степень ограничена длиной максимального слова заданного языка.

[6] При этом очень важно отметить *связь* вариантов *метрики* с вариантами *частичного порядка* – в нашем случае на множестве языков. Подробнее см. в процитированных выше наших предыдущих публикациях.

- либо будем иметь в виду «построение любого» (корня из языка либо инверсного морфизма)[7].

Экспоненциальный алгоритм для любого варианта задачи извлечения корня очевиден: надо просто перебрать все подмножества множества потенциальных корней, и среди этих подмножеств выбрать подходящее (подходящие). Поэтому проблема состоит в описании возможных полиномиальных алгоритмов для этих вариантов задачи.

III. СЛУЧАЙ 1-БУКВЕННОГО АЛФАВИТА И КВАДРАТНОГО КОРНЯ: ПРОСТЫЕ МОДЕЛИ РЕШЕНИЯ ЗАДАЧИ

Всюду далее будем считать, что на основе входных данных мы получили N потенциальных корней – повторим, что все потенциальные корни можно найти с помощью несложного полиномиального алгоритма. Из названия раздела понятно, что здесь мы будем пытаться искать решение самого простого варианта задачи: нужно извлечь квадратный корень (корень 2-й степени), при том, что алфавит состоит из 1 буквы; но, как уже было отмечено, задача даже в таком случае не является простой. Мы для такого «простого» варианта не знаем полиномиального алгоритма – поэтому для решения задачи пытаемся применять различные подходы; при этом, как также следует из названия раздела, здесь мы рассмотрим подходы самые простые, они, по-видимому, сразу понятны из описания задачи.

Будем работать с множеством потенциальных корней. И, поскольку каждый корень определяется своей длиной, то не ограничивая общности мы будем считать, что эта длина находится в пределах от 0 до некоторого n – так и было предположено ранее, обозначение то же самое; более того, в рассматриваемых далее примерах всегда потенциальный корень 0 имеется[8].

Расположим выбранные значения от 0 до n в виде пометок строк и столбцов таблицы $(n+1)\times(n+1)$, при этом в клетки таблицы записываем сумму чисел – пометок соответствующих строки и столбца. В клетках таблицы отметим значения, входящие в исходное множество (на конкретном примере, приведённом на рис. 1, отмечены звёздочками), также отметим значения потенциальных корней (на рисунке отмечены зелёным фоном).
Конечно, на каждой из диагоналей (называем таковыми те, которые перпендикулярны главной диагонали) все элементы соответствуют одному и тому же значению.

Понятно, что задачу извлечения квадратного корня можно переформулировать так: чтобы выбрать среди «зелёных» элементов некоторое подмножество[9], такое, что:

- пересечение каждой пары выбранных элементов отмечено (звёздочкой);

	0	1	2	3	4	5	6
0	*	*	*	*	*		*
1	*	*	*	*		*	*
2	*	*	*		*	*	
3	*	*		*	*		*
4	*		*	*		*	
5		*	*		*		
6	*	*		*			*

Рис. 1. Первый вариант модели для квадратного корня в случае 1-буквенного алфавита. Пример для исходного языка $[0\,1\,3\,6]$.

- на каждой из диагоналей присутствует хотя бы один элемент, входящий в такую пару.

Однако при такой переформулировке среди сведе́ний этой задачи очевидным представляется только одно – сведе́ние к уже упомянутой задаче о покрытии множества.

Более того. Здесь можно увидеть большую аналогию с рассматривавшейся нами во многих публикациях задачей минимизации недетерминированных конечных автоматов (см. [11], [12], [14], [15], [16], [17], [18], [19], [21] и мн. др.): в решаемой здесь задаче извлечения корня требования к выбираемому подмножеству потенциальных корней практически совпадают с требованиями на грид (блок) для минимизации автоматов. При этом отдельно отметим статьи, связанные с языком (автоматом) Ватерлоо, [11], [13], [16], [17], [20], [21] и др. Таким образом, в рассматриваемой модели мы получаем бинарное отношение – но, в отличие от задачи минимизации автоматов, определённое на совпадающих множествах. Таким же образом определённые бинарные отношения можно рассматривать в качестве матриц смежности соответствующих графов – что мы и будем делать ниже.

Рассмотрим рис. 1 немного подробнее. Для него:

- язык, взятый для формирования примера, – $[0\,1\,3\,6]$;

тогда:

- его квадрат (считаем именно его исходным языком для этой задачи) –

$$[0\ 1\ 2\ 3\ 4\ 6\ 7\ 9\ 12]; \tag{3}$$

- потенциальные корни – $[0\ 1\ 2\ 3\ 6]$.

Легко показать, что потенциальные корни 0 и 6 в ответ обязательно входят – как минимальный и максимальный по длине.

Всё приведённое выше (с изменениями, соответствующими конкретной задаче) может быть применено к любой задаче извлечения квадратного корня – однако дальнейшие рассуждения относятся только к задаче рассматриваемой. В ней 1 входит в ответ по той причине, что без него в исходном языке (3) не может быть значения 1, а 3 входит по аналогичной причине. Однако 2 входить в ответ не может – например, потому, что элементом этой матрицы $(2, 3)$ не является звёздочка.

Таким образом, *для этой корнкретной задачи* удаётся доказать, что язык $[0\,1\,3\,6]$, взятый для формирования примера, является единственным возможным ответом. Однако в общем случае соответствующего «алгоритма»

[7] Т. е., по-видимому, самую простую (в сравнении с остальными) задачу.

[8] При этом, аналогично большинству смежных задач, рассматривавшихся в наших предыдущих публикациях, мы здесь можем «сократить» задачу – т. е. *поделить* все значения на минимальное слово. Очень простой подтверждающий пример – таков: из исходного языка $[2\,3\,4\,5\,6\,8]$ извлекается (квадратный) корень $[1\,2\,4]$ – а «деление» последнего языка на 1 даёт язык $[0\,1\,3]$, квадрат которого равен $[0\,1\,2\,3\,4\,6]$, т. е. квадрат просто равен исходному языку, «делённому» на 2.

[9] Конечно же, одно и то же для «зелёных» строк и «зелёных» же столбцов.

у нас нет. И, как мы уже отмечали, очевидным сведéнием задачи является только рассмотрение задачи о покрытии множества. А «улучшить ситуацию» (т. е. получить описание полиномиального алгоритма) нам пока не удалось – однако последующие модели представляются достаточно интересными.

Следующая модель является некоторой модификацией предыдущей. Здесь строки и столбцы таблицы помечены только потенциальными корнями (что можно считать логичным: ведь только из них выбираются подмножества-ответы), и при этом на пересечении вместо звёздочек можно поставить сумму. Но мы числовые значения возможных сумм использовать не будем, поскольку, по-видимому, нагляднее заменять их на какие-то другие знаки [10] – что мы и делаем.

В примере, рассматриваемом для того же исходного языка (3), мы вместо 9 элементов этого языка проставляем в таблицы буквы A B … H I; результат получаем в таблице, приведённой на рис. 2.

	0	1	2	3	6
0	A	B	C	D	F
1	B	C	D	E	G
2	C	D	E		
3	D	E		F	H
6	F	G		H	I

Рис. 2. Второй вариант модели для предыдущего примера.

При этом несколько иной является и *цель*. Мы также выбираем некоторое подмножество потенциальных корней (и в этой модели они уже понятны, «зелёный цвет» не нужен), а отличия от модели предыдущей таковы:

- пересечение каждой пары выбранных элементов отмечено (некоторой буквой);
- любая из использованных букв входит в хоть одну такую пару.

Понятно, что результат решения задачи – тот же самый, однако, как нам кажется, здесь он виден сразу. (То есть мы считаем, что описание соответствующих алгоритмов поиска подмножества множества потенциальных корней и реализация этих алгоритмов в виде компьютерных программ существенно проще.)

В качестве последней простой модели рассмотрим граф, который может быть построен на основе любой из моделей предыдущих (конечно,более подходящей является вторая модель). Вершины графа помечены потенциальными корнями, а рёбра – буквами, совпадающими с приведёнными в таблице на рис. 2. Также считаем, что все вершины имеют принадлежащие им петли – мы не будем их рисовать, но будем ставить около всех вершин ещё и соответствующие этим петлям буквы, которые совпадают с буквами, помещёнными на главную диагональ таблицы. Таким образом, для нашего примера получаем граф, приведённый на рис. 3.

Цель обработки получаемого графа такова. Необходимо выбрать клику (не обязательно максимально возможного размера), содержащую все использованные буквы;

[10] Такой вариант будет более удобен в следующей модели.

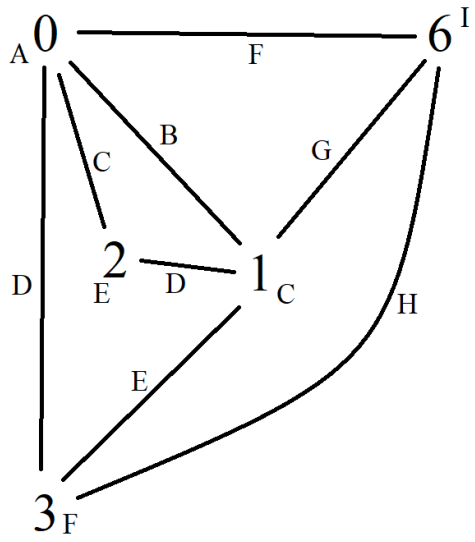

Рис. 3. Графовая модель для рассматриваемого примера.

вершины клики и дадут подмножество множества потенциальных корней, являющееся ответом. Понятно, что для рассматриваемого примера единственный возможный ответ – тот же самый, что и выше: подмножество множества вершин $\{0, 1, 3, 6\}$ кликой является, причём, конечно, единственной кликой, удовлетворяющей сформулированному дополнительному условию.

В заключение раздела укажем несколько ссылок на книги по теории графов, с которыми согласованы названия применяемых в статье понятий: [10], [22], [23], [24], [25], [26], [27].

А весь дальнейший материал статьи можно – *несколько всё упрощая* – охарактеризовать так: мы будем рассматривать *дополнение* графа, рассмотренного в конце предыдущего раздела. При этом необходимо сделать такие два примечания.

- Во-первых, хотя мы и говорим про графы, собственно «графовую» терминологию мы будем употреблять мало; однако нередко она будет применяться *неявно*.
- Во-вторых, варианты возможных ответов, находимые в результате описанных в следующих разделах моделей, нужно будет считать решениями «предварительными»: все они будут нуждаться в дальнейшей проверке.

IV. СЛУЧАЙ 1-БУКВЕННОГО АЛФАВИТА И КВАДРАТНОГО КОРНЯ: ТАБУИРОВАННЫЕ ПАРЫ И ТАБУИРОВАННЫЕ ГИПЕРПЛОСКОСТИ

В этом разделе мы снова будем считать, что на основе входных данных получены N потенциальных корней – и ещё раз повторим, что все потенциальные корни можно найти с помощью несложного полиномиального алгоритма. Для некоторого упорядочивания этих потенциальных корней (естественным упорядочиванием в случае 1-буквенного алфавита является возрастание длины потенциального корня) рассмотрим N-мерный куб, у которого N двоичных координат каждой вершины имеют следующий смысл:

- 0 – соответствующий потенциальный корень в предполагаемый корень-ответ не входит;
- 1 – входит.

Понятно, что ответ на любую из задач извлечения корня [11] можно получить путём перебора вершин такого N-мерного куба – причём каждая проверка тривиальным образом осуществляется за полиномиальное время. Однако в связи с тем, что общее число вершин куба равно 2^N, все подобные итоговые алгоритмы извлечения корня получаются экспоненциальными.

Для дальнейшего будем рассматривать пары потенциальных корней; заранее отметим, что перебор всех пар полиномиален (осуществляется тривиальным квадратичным алгоритмом). Поскольку каждая такая пара представляет собой два потенциальных корня, мы можем сказать, является ли их произведение допустимым (иными словами – входит ли сумма их длин в заданное слово). Если произведение допустимым не является – то такую пару потенциальных корней будем называть табуированной (табу-парой).

Каждая такая пара в рассматриваемом N-мерном кубе определяет N−2-мерную гиперплоскость (будем называть каждую из них табу-плоскостью) – и все элементы этой гиперплоскости заведомо *не* являются корнями. Однако, конечно, только на основе этого факта решить задачу извлечения корня нельзя: например, вершина N-мерного куба, все координаты которой равны 0, по определению не входит ни в одну табу-плоскость (поскольку у всех вершин всех табу-плоскостей имеются по крайней мере две равные 1 координаты) – но при этом ответом может являться только в вырожденных случаях. Но, несмотря на приведённый пример, название «табу-плоскость» полностью соответствует применению этих гиперплоскостей в рассматриваемой нами задаче.

Применение множества всех табу-плоскостей понятно. А именно, на основе дальнейшего материала настоящей статьи (конкретно – на основе раздела VI) можно показать, что возможно некоторое улучшение алгоритма поиска (хотя бы одного) корня: достаточно перебрать только такие вершины N-мерного куба (будем считать, что вершина – это $B = (b_1 b_2 \dots b_N)$, а сами исходные данные – это $A \in 2^{2n}$), которые:

- сами не входят ни в одну табуированную плоскость; т. е.
$$\left(\forall i, j \in \overline{1,n},\ i \neq j\right)\left(\widehat{\#i,\#j} \notin \mathcal{T}(A)\right);$$
- обладают свойством
$$\left(\exists i \in \overline{1,n},\ b_i = 0\right)\left(\exists j \in \overline{1,n},\ b_j = 1\right)$$
$$\left(\widehat{\#i,\#j} \in \mathcal{T}(A)\right) -$$
т. е. при изменении значения какой-то координаты с 0 на 1 (i-й координаты в приведённой записи) предыдущее условие нарушится;

только среди таких вершин n-мерного гиперкуба и надо проверить выполнение условия $B^2 = A$.

Рассмотрим конкретную задачу – 1-буквенный алфавит, 2-я степень, на входе – квадрат языка
$$B = [0\ 1\ 3\ 6]$$
(специально отметим, что этот язык B при формулировке задачи неизвестен), т. е. язык
$$A = B^2 = [0\ 1\ 2\ 3\ 4\ 6\ 7\ 9\ 12].$$
Потенциальные корни в нашем примере –
$$B = [0\ 1\ 2\ 3\ 6],$$
а множество табуированных пар –
$$\mathcal{T} = \left\{\widehat{2,3},\ \widehat{2,6}\right\} = \left\{\widehat{\#3,\#4},\ \widehat{\#3,\#5}\right\}.$$

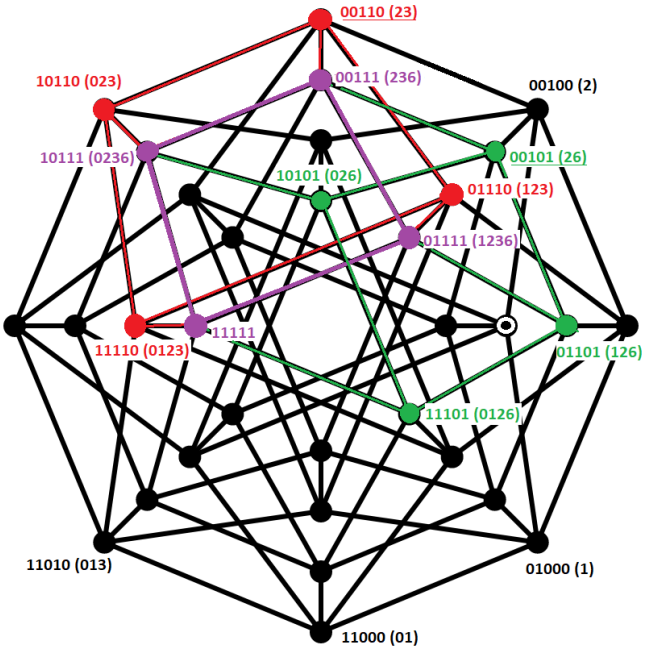

Рис. 4. Граф с раскрашенными вершинами и рёбрами, отражающими варианты включения потенциальных корней. Граф выполнен в форме многомерного куба. Здесь – пример 5-мерного куба для конкретной исходной задачи: [0 1 3 6]; квадрат: [0 1 2 3 4 6 7 9 12]; 5 потенциальных корней: [0 1 2 3 6]; имеются 2 табуированные пары, причём содержащие общий потенциальный корень.

Рассмотрим следующий рисунок 5-мерного гиперкуба (рис. 4). Приведём к рисунку несколько комментариев.

- Для наглядности на рисунке помечены не все из 32 вершин куба (чтобы не «перегружать» рисунок информацией) – но пометки остальных могут быть определены на основе пометок приведённых. Пометки – 5-значные двоичные числа (содержащие при необходимости незначащие 0 в старших разрядах); они записаны до обозначений в скобках (о которых ниже).
- При этом «начало координат» двоичным кодом специально не помечено (также для того, чтобы не «перегружать» рисунок); это – вершина, нарисованная с чёрным внутренним кружком.
- В скобках – после двоичной записи обозначения вершины гиперкуба – без пробелов записываем значения соответствующих потенциальных корней; это – сами значения, а не их номера (т. е. в рассматриваемом примере – 2, а не #3), причём выбираем только те значения, которые соответствуют 1 в двоичной записи вершины гиперкуба.
- Красным цветом показана гиперплоскость (фактически – обычный 3-мерный куб), соответствующая табуированной паре $\widehat{2,3}$.

[11] Не только рассматриваемую в первую очередь самую простую задачу – когда в алфавите 1 буква, а степень равна 2.

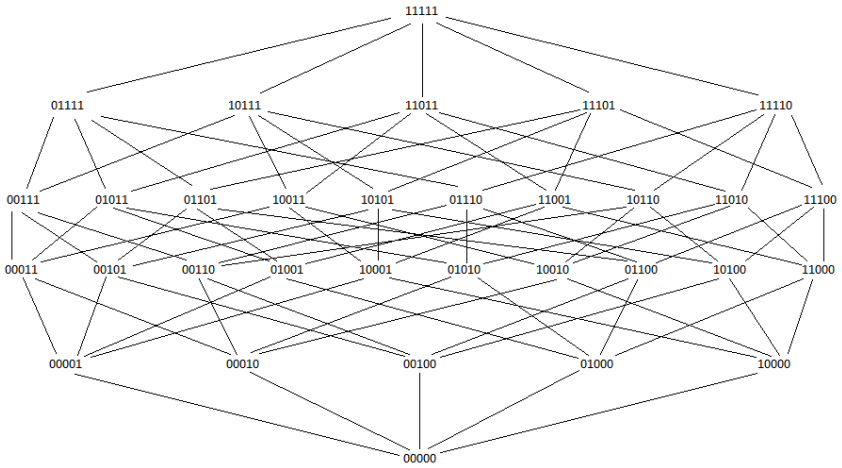

Рис. 5. Граф булеана общего вида размерности 5.

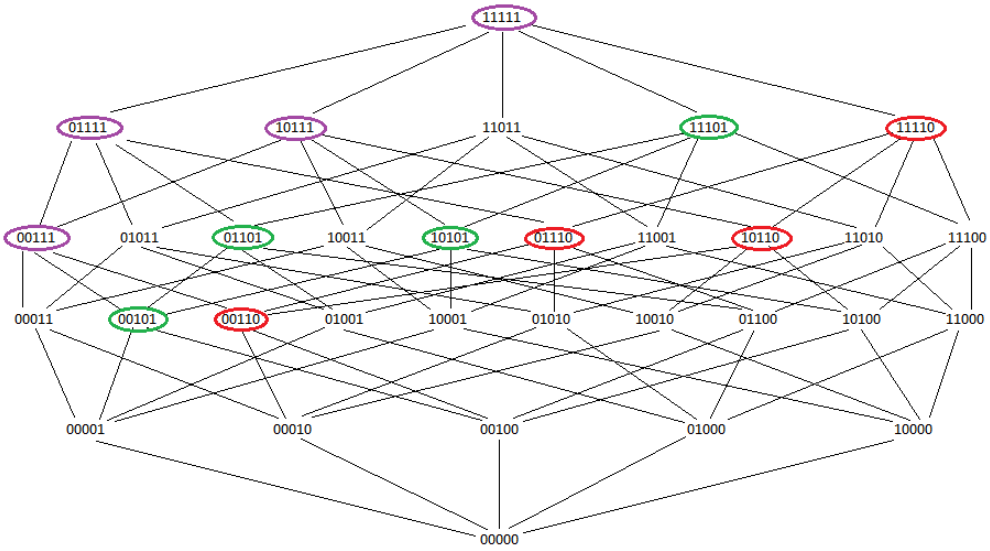

Рис. 6. Граф булеана размерности 5 для рассматриваемой задачи [0 1 3 6]. Использованы те же самые цветы, что и на рис. 4. Фактически граф изоморфен графу, приведённому на рис. 4, причём с учётом отмеченных разными цветами вершин.

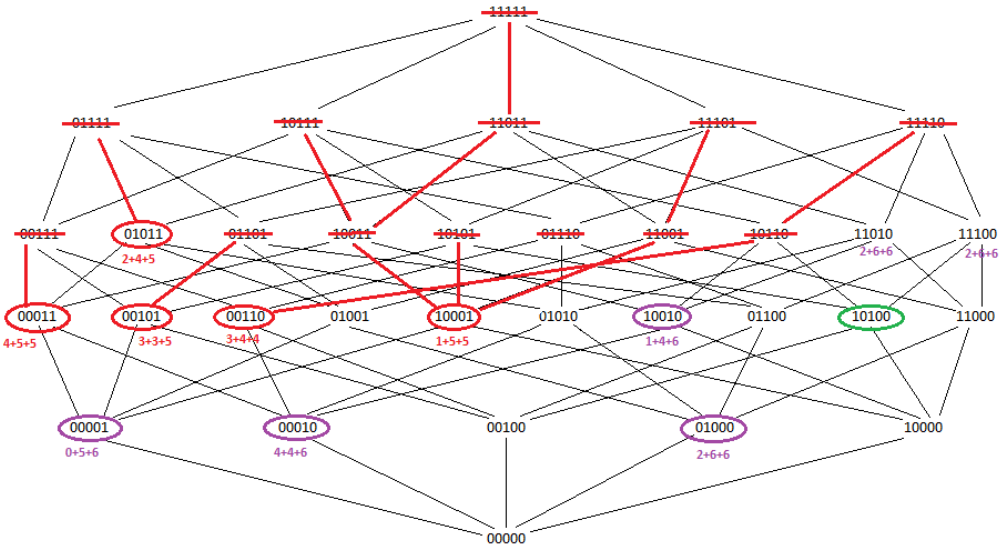

Рис. 7. Подграф графа булеана (размерность 5 из 7) для рассматриваемой задачи [0 1 3 6] и 3-й степени. Из 7 потенциальных корней рассмотрены 5: (1 2 3 4 5) (исключили самый короткий и самый длинный, они заведомо входят в любое решение). Поэтому, 10011 соответствует множеству [0 1 4 5 6]. Смысл использования цветов – иной, для красного и зелёного цветов он соответствует рис. 8, подробности в тексте статьи.

- При этом «якорная» вершина гиперплоскости – т. е. та вершина, в которой значения 1 имеются только у двух координат, эту гиперплоскость определяющих, – подчёркнута.
- Аналогично, зелёным цветом показана гиперплоскость, соответствующая табуированной паре $\overbrace{2,6}$.
- Пересечение этих двух гиперплоскость (фактически – обычный квадрат), показано фиолетовым цветом. Конечно, этому пересечению принадлежит максимальная точка гиперкуба.

И, конечно, можно рассматривать такой гиперкуб в виде булеана; более того, работать с последним удобнее – а булеан принято рисовать иначе, что мы и сделаем. Изображение «в виде булеана» удобнее тем, что из любой его табуированной вершины можно проводить «рёбра вверх», отмечая при этом новые вершины (в которые мы при этом попадаем) также как табуированнные. Но, повторим, эти 2 варианта графов изоморфны – и на рис. 6 приведён граф булеана для рассматриваемой задачи.

Итак, будем рассматривать булеан, соответствующий N-мерному кубу. На рис. 5 приведён граф булеана общего вида размерности 5; при этом на нём, как и при его же изображении в виде гиперкуба, можно отмечать табуированные плоскости.

V. ОБОБЩЕНИЕ ЗАДАЧИ ДЛЯ ПРОИЗВОЛЬНОГО АЛФАВИТА

Когда мы говорим про алфавит произвольной мощности, то, вероятно, можно считать, что число букв равно 2: по-видимому, все *содержательные* примеры можно «закодировать» 2 буквами. В этом разделе мы будем рассматривать только пример – для которого просто поясним все понятия, рассматривавшиеся выше (т. е. переопределим аналогичные понятия для этого алфавита $\Sigma = \{\mathtt{a}, \mathtt{b}\}$); отметим, что пример имеет много общего с некоторыми примерами, рассматривавшимися в некоторых наших предыдущих публикациях.

Пусть:

- язык, взятый для формирования примера, –

$$B = \{\mathtt{a}, \mathtt{bab}\};$$

тогда:

- его квадрат (как и в предыдущих примерах, считаем именно его *исходным языком* для этой задачи) –

$$A = B^2 = \{\mathtt{aa}, \mathtt{abab}, \mathtt{baba}, \mathtt{babbab}\}; \qquad (4)$$

- потенциальные корни –

$$\{\mathtt{a}, \mathtt{ab}, \mathtt{ba}, \mathtt{bab}\} = B \cup \{\mathtt{ab}, \mathtt{ba}\}.$$

Аналогично предыдущим примерам, легко показать, что потенциальные корни a и bab в ответ обязательно входят – как минимальный и максимальный по длине; однако, чтобы показать общий принцип работы с булеаном, мы не будем пользоваться этим фактом.

В связи с последним рассмотрим булеан размерности 4; в нём будем применять алфавитный порядок потенциальных корней:

- № 1: a,
- № 2: ab,
- № 3: ba,
- № 4: bab.

В отличие от 1-буквенного алфавита (для которого выполняется коммутативность произведения: каждое произведение некоторых двух потенциальных корней, вообще говоря, не зависит от их порядка в произведении), при работе с алфавитами большей мощности [12] каждая точка булеана (при желании использовать его для описании модели) соответствует 2 вариантам произведения потенциальных корней; поэтому на рисунке булеана (рис. 8 в рассматриваемом примере) будем ставить красные точки в те вершины, для которых *хотя бы одно из двух* таких произведений табуировано (т. е. не входит в исходный язык).

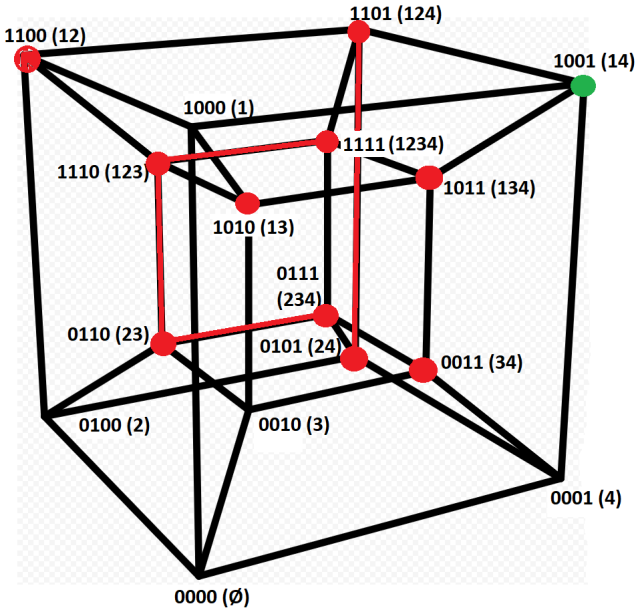

Рис. 8. Подграф графа булеана для примера $(\mathtt{a\ bab})^2$. Номера потенциальных корней взяты по алфавиту: № 1: a, № 2: ab, № 3: ba, № 4: bab. Смысл использования цветов – тот же, что и на рис. 7, подробности в тексте статьи.

Поэтому для приведённого на рис. 8 графа булеана помечаем красным следующие вершины:

- $\overbrace{1,2}$ (вершина 1100) – поскольку [13] $\mathtt{aab} \notin A$;
- $\overbrace{1,3}$ (вершина 1010) – поскольку $\mathtt{aba} \notin A$;
- $\overbrace{2,3}$ (вершина 0110) – поскольку $\mathtt{abba} \notin A$;
- $\overbrace{2,4}$ (вершина 0101) – поскольку $\mathtt{abbab} \notin A$;
- $\overbrace{3,4}$ (вершина 0011) – поскольку $\mathtt{babab} \notin A$.

Отметим, что пару $\{1,4\}$ мы табуированной не считаем: конечно, это очевидно (именно так и создавался пример) – однако при решении конкретной задачи извлечения корня на вход поступает только язык A, и «отсутствие табу» для каждой подобной пары приходится проверять путём рассмотрения двух соответствующих

[12] Все возможные «контрпримеры» для них (т. е. примеры комутирования) могут быть описаны одним равенством,

$$u^i \cdot u^j = u^j \cdot u^i,$$

выполняющемся для любого слова u и для любых $i, j \geqslant 0$; этот факт несложно доказывается в различных книгах по теории формальных языков уровня . Например, $\mathtt{ab} \cdot \mathtt{abab} = \mathtt{abab} \cdot \mathtt{ab}$.

[13] Здесь и далее при наличии нескольких возможных объяснений для «табуированности» некоторой вершины булеана мы приводим только одно из них.

конкатенаций. Помечаем эту пару $\{1,4\}$ зелёным цветом (подходящая) – и забегая вперёд отметим, что для решения задачи достаточно найти «зелёную вершину», лежащую ближе всего к максимальной вершине $(1,1,\dots 1)$.

Далее. Как и ранее, тройки потенциальных корней табуированы тогда и только тогда, когда в тройку как подмножество входит какая-либо табуированная пара. Поэтому помечаем красным такие тройки – вместе с указанием некоторой сответствующей тройке пары, это соответствие отмечаем красным отрезком. Более детальные подробности не описываем, см. рисунок, на нём согласно такому же правилу отмечена и единственная четвёрка (полное множество, максимальный элемент булеана).

Самая близкая к максимальной вершине вершина «зелёная» – это 1001; она и соответствует единственно возможному ответу (единственному корню из заданного языка), корни №2 ab и №3 ba в ответ не вошли. Впрочем, рассматриваемая задача была сформирована так, что именно это и ожидалось.

VI. О ДОБАВЛЕНИИ НОВОЙ КООРДИНАТЫ К УЖЕ ИМЕЮЩЕМУСЯ РЕШЕНИЮ

Материал этого раздела описывает ту часть предмета статьи, которая, по-видимому, является единственной нетривиальной. Именно на основе результатов доказываемой далее теоремы мы можем утверждать, что поиск (хотя бы одного) корня может производиться как поиск вершины гиперкуба (булеана), ближе всего находящейся к максимальной вершине и при этом не входящей ни в одну из табуированных плоскостей.

Специально отметим, что на основе материала предыдущего раздела можно будет также утверждать, что таким же образом задачу можно решать и для произвольного алфавита.

Теорема 1: Пусть

$$B = (b_1 b_2 \dots b_N) \in 2^N \quad -$$

некоторый корень уравнения (1), $M = 2$. Пусть при этом

$$b_k = 0 \quad \text{для некоторого} \quad k \in \overline{1,N}.$$

Пусть также для каждого $i \in \overline{1,N} \setminus \{k\}$, такого что $b_i = 1$, выполнено условие

$$\widehat{k,i} \notin \mathcal{T}(A).$$

Тогда

$$B_{+k} = (b_1 b_2 \dots b_N)_{+k}$$

также является корнем уравнения (1).

Доказательство. Очевидно, что

$$B^n \subseteq (B_{+k})^n.$$

Поэтому если бы было выполнено неравенство

$$(B_{+k})^n \neq A,$$

то существовал бы разряд $i \in \overline{1,N} \setminus \{k\}$, такой что $b_i = 1$, и при этом при возведении в квадрат у произведения появлялся бы *новый* разряд, равный 1. Поскольку мы добавили только один, k-й разряд, к B, последний факт возможен только при

$$\widehat{k,i} \in \mathcal{T}(A) \; -$$

что противоречит условию теоремы. □

Итак, именно на основе результатов этой теремы можно утверждать, что для поиска хоть какого-то решения исходной задачи можно искать «нетабуированные» точки булеана, лежащие ближе всего к максимальной вершине. Решение – если хотя бы одно имеется – обязательно будет среди тех «нетабуированных» точек, для которых не существует соседней «нетабуированной», лежащей приэтом ближе к максимальной вершине.

VII. БОЛЕЕ СЛОЖНЫЕ ПРИМЕРЫ

В этом разделе мы рассмотрим менее тривиальные примеры, чем рассматривались ранее.

В первом примере интересны:

- бóльшая (чем ранее) размерность;
- несвязность подграфа графа булеана, описывающего рассматриваемую задачу.

Этого в предыдлущих примерах не было.

Итак, пусть:

- язык, взятый для формирования примера, – 6855; в других вариантах записи его же –

  [0 1 2 6 7 9 11 12] и 1101011000111;

  (малые пробелы здесь и далее добавляем в каждой двоичной записи через каждые 10 позиций)

тогда:

- его квадрат (считаем именно его исходным языком для этой задачи) –

  11111 1111111111 1111011111 (33554399);

- потенциальные корни –

  [0 1 2 3 4 5 6 7 8 9 10 11 12]

  (т. е. все возможные значения в заранее известном промежутке от 0 от 12), в другой записи – 8191.

Как и ранее, потенциальные корни 0 и 12 в ответ обязательно входят – как минимальный и максимальный по длине.

Итак, всего потенциальных корей – 13. Понятно, что на расстоянии 0 от максимальной вершины булеана корня нет (хотя бы потому, что в ответе имеется разряд, равный 0). Кроме того, для подобных примеров[14] разряды 0 у квадрата могут появляться:

- только в качестве первого[15] или последнего разряда – при наличии в возводимом в квадрат значении только одного 0;
- не далее третьего разряда – при наличии в этом значении двух 0.

Оба этих факта несложно доказываются средствами элементарной математики. Таким образом, на расстояниях 0, 1 и 2 от максимальной вершины 13-мерного куба корней быть не может.

А вот на расстоянии 3 корни есть – и все эти корни (их 21) могут быть найдены с помощью несложной компьютерной программы.

[14] С достаточным числом 1 в двоичной записи; конкретное число этих 1 можно вычислить – однако очевидно, что здесь их действительно «достаточно».

[15] По номеру – 0-го: выше уже было отмечено, что мы для этого представления данных нумеруем разряды начиная с 0 – аналогично программам на Си-подобных языках.

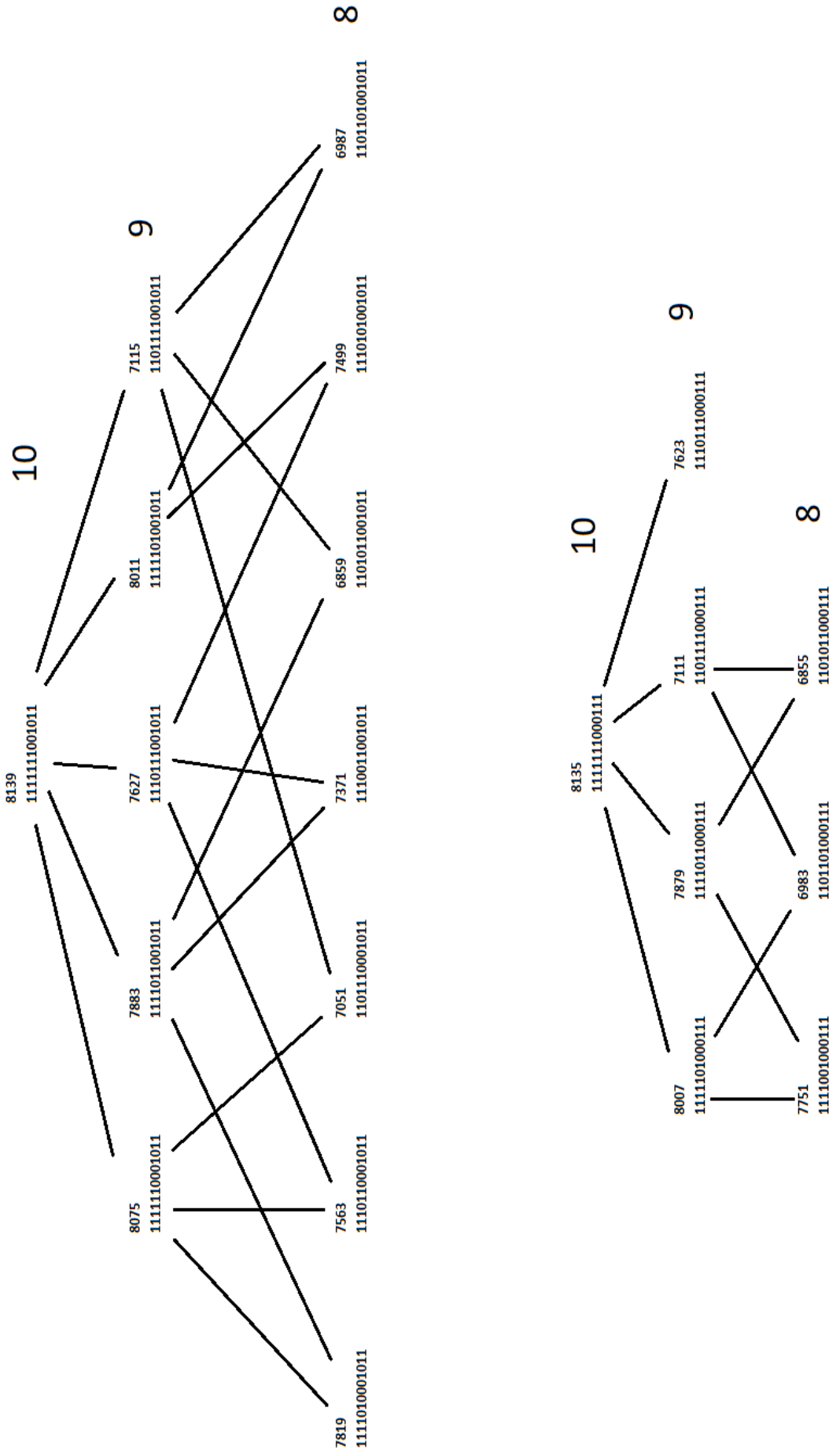

Рис. 9. Подграф графа булеана размерности 13 со всеми корнями квадрата слова 110 1011000111 [0 1 2 6 7 9 11 12] (6855). Справа от элементов (если смотреть на рисунок «правильно», сбоку) показаны уровни булеана – количества потенциальных корней.

На рис. 9 все эти корни показаны как вершины графа – являющегося подграфом стандартного изображения булеана. Справа от элементов показаны уровни булеана – количества потенциальных корней. Как видно из рисунка, все корни расположены на уровнях 10, 9 и 8 (т. е. на расстояниях 3, 4 и 5 от максимальной вершины).

Теперь рассмотрим пример, когда (квадратные) корни есть – но ближайший к максимальной вершине корень находится от неё на расстоянии 4. По-видимому, минимальный такой пример – следующий [16].

- Язык, взятый для формирования примера, – 339; в других вариантах записи его же –

$$[0\ 2\ 4\ 7\ 8] \quad \text{и} \quad 101010011;$$

тогда:

- его квадрат (считаем именно его исходным языком для этой задачи) –

$$1110101\,1111110111 \qquad (120823);$$

- потенциальные корни –

$$[0\ 1\ 2\ 3\ 4\ 5\ 6\ 7\ 8]$$

(т. е. снова все возможные значения в заранее известном промежутке, здесь – от 0 от 8), в другой записи – 511.

Простым перебором на компьютере можно убедиться, что нет корней на расстояниях от максимальной вершины:

- 0 – для этого необходимо проверить 1 вариант;
- 1 – для этого необходимо проверить 9 вариантов; при этом применив несложные рассуждения [17], можно ограничить проверку 4 вариантами;
- 2 – для этого необходимо проверить $C_9^2 = 36$ вариантов; аналогично предыдущему пункту, после несложных рассуждений можно ограничить проверку 6 вариантами (C_4^2);
- 3 – для этого необходимо проверить $C_9^3 = 56$ вариантов; аналогично предыдущему, после несложных рассуждений можно ограничить проверку 4 вариантами (C_4^3);

Повторим, что применяя несложные предварительные рассуждения, можно существенно ограничить число перебираемых вариантов – и осуществить необходимую проверку без компьютера.

Итак, на расстоянии 3 и менее от максимальной вершины корней нет – однако способ формирования квадрата (исходного значения) показывает, что на расстоянии 4 корень есть.

Подобным же образом можно формировать примеры, в которых корни есть, и при этом ближайший корень находится от максимальной вершины на сколь угодно большом (но, конечно, заданном заранее) расстоянии. Однако, по-видимому, способ такого формирования, как и доказательство корректоности соответствующего алгоритма, не представляют большого интереса для темы настоящей статьи.

[16] Минимальный с точки зрения соответствующего значения, взятого как исходное для возведения в квадрат.

[17] То есть фактически пример “data preprocessing”. Не будем описывать эти рассуждения подробно: в любом случае, можно очень легко осуществить необходимый перебор с помощью компьютера.

VIII. ЗАКЛЮЧЕНИЕ

Опишем план ближайшей запланированной статьи-продолжения, а также возможные направления дальнейших работ по этой тематике.

Сначала без подробных комментариев приведём рисунок 10, описывающий попытку обобщения нескольких рассмотренных выше моделей на случай корня произвольной степени. В ближайшей статье мы предполагаем:

- подробнее – чем в приведённой подписи к рисунку – его прокомментирвать;
- пояснить, в связи с чем такое обобщение возможно.

Однако заранее отметим, что это обобщение приводит к достаточно сложным для программной реализации моделям.

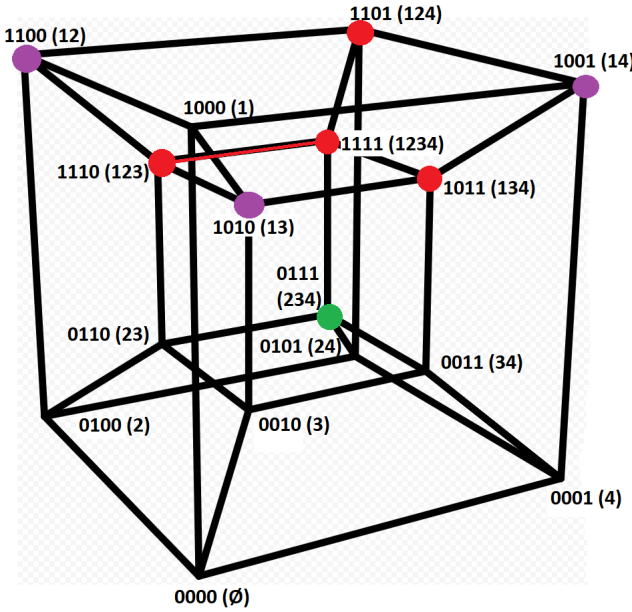

Рис. 10. Подграф графа булеана для примера $(\mathrm{aba\ ba\ b})^3$. Номера потенциальных корней взяты по алфавиту: №1: ab, №2: aba, №3: b, №4: ba, они совпадают с выбранными на рис. 8. Смысл использования цветов – тот же, что и на рис. 7.

Также в ближайшей статье мы планируем привести формулировку эвристического алгоритма решения задачи в простейшем случае (1-буквенный алфавит, корень 2-й степени). Алгоритм основывается на ещё не доказанной нами гипотезе, связанной с т. н. *важными* потенциальными корнями [18], – и во всех известных нам случаях работает правильно.

Также в будущих публикациях мы предплагаем описать свéдение некоторых вспомогательных подзадач, получающихся при решении задачи извлечения корня 2-й степени, к проблеме 2–SAT, [9] и др.; этот метод применим и в случае, когда $M = 2^m$ для некоторого $m \geqslant 2$. При этом мы заранее отмечаем, что для задачи извлечения корня 3-й степени возможно подобное же свéдение к проблеме 3–SAT – в связи с чем, по-видимому, для корней, степень которых не является степенью числа 2, подобное направление возможного решения задачи вряд ли приведёт к успеху.

[18] Схожее понятие рассматривалось в одной из наших предыдущих публикаций; более подробную информацию и ссылки здесь приводить не будем.

IX. БЛАГОДАРНОСТИ

Работа первого автора частично поддержана грантом научной программы китайских университетов “Higher Education Stability Support Program” (раздел “Shenzhen 2022 – Science, Technology and Innovation Commission of Shenzhen Municipality”).

Список литературы

[1] Саломаа А. *Жемчужины теории формальных языков*. – М., Мир. – 1986. – 159 с.

[2] Narendran B., Tiwari P. *Polynomial root-finding: analysis and computational investigation of a parallel algorithm* // In: Proceedings of ACM Symposium on Parallelism in Algorithms and Architectures (1992), SPAA’92.

[3] Melnikov B., Korabelshchikova S., Dolgov V. *On the task of extracting the root from the language* // International Journal of Open Information Technologies. – 2019. – Vol. 7. No. 3. – P. 1–6.

[4] Мельников Б. *Полурешётки подмножеств потенциальных корней в задачах теории формальных языков. Часть I. Извлечение корня из языка* // International Journal of Open Information Technologies. – 2022. – Vol. 10. No. 4. – P. 1–9.

[5] Гинзбург С. *Математическая теория контекстно-свободных языков*. М., Мир. – 1970. – 326 с.

[6] Ахо А., Ульман Дж. *Теория синтаксического анализа, перевода и компиляции. Т. 1*. М., Мир. – 1978. – 613 с.

[7] Хопкрофт Дж., Мотвани Р., Ульман Дж. *Введение в теорию автоматов, языков и вычислений*. М., Вильямс. – 2008. – 528 с.

[8] Громкович Ю. *Теоретическая информатика. Введение в теорию автоматов, теорию вычислимости, теорию сложности, теорию алгоритмов, рандомизацию, теорию связи и криптографию*. СПб., БХВ-Петербург. – 2010. – 336 с.

[9] Гэри М., Джонсон Д. *Вычислительные машины и трудно-решаемые задачи*. М., Мир. – 1982. – 416 с.

[10] Еремеев А., Заозерская Л., Колоколов А. *Задача о покрытии множества: сложность, алгоритмы, экспериментальные исследования* // Дискретный анализ и исследование операций, сер. 2. – 2000. – Vol. 7. No. 2. – P. 22–46.

[11] Kameda T., Weiner P. *On the state minimization of nondeterministic finite automata* // IEEE Transactions on Computers. – 1970. – Vol. C-19, No. 7. – P. 617–627.

[12] Jiang T., Ravikumar B. *Minimal NFA problems are hard*. SIAM Journal on Computing. 1993, Vol. 22, No. 6, P. 1117–1141.

[13] Мельников Б. *Подклассы класса контекстно-свободных языков (монография)*. – М., Изд-во Московского университета. – 1995. – 174 с. – ISBN 5-211-03448-1.

[14] Melnikov B., Vakhitova A. *Some more on the finite automata* // The Korean Journal of Computational and Applied Mathematics (Journal of Computational and Applied Mathematics). – 1998. – Vol. 5, No. 3. – P. 495–505.

[15] Melnikov B. *A new algorithm of the state-minimization for the nondeterministic finite automata*. The Korean Journal of Computational and Applied Mathematics (Journal of Applied Mathematics and Computing). 1999, Vol. 6, No. 2. – P. 277–290.

[16] Polák L. *Minimalizations of NFA using the universal automaton* // International Journal of Foundation of Computer Science. – 2005. – Vol. 16, No. 5. – P. 999–1010.

[17] Lombardy S., Sakarovitch J. *The universal automaton* // Logic and Automata. Amsterdam University Press. – 2008. – P. 457–504.

[18] Hromkovič J., Petersen H., Schnitger G. *On the limits of the communication complexity technique for proving lower bounds on the size of minimal NFA’s*. Theoretical Computer Science. 2009. – No. 410. – P. 2972–2981.

[19] Yo-Sub Han. *State elimination heuristics for short regular expressions*. Fundamenta Informaticae. 2013. – No. 128. – P. 445–462.

[20] Долгов В., Мельников Б. *Об алгоритмах автоматического построения Ватерлоо-подобных конечных автоматов на основе полных автоматов* // Эвристические алгоритмы и распределенные вычисления. – 2014. – Т. 1. № 4. – С. 24–45.

[21] Мельников Б. *Регулярные языки и недетерминированные конечные автоматы (монография)*. – М., Изд-во Российского государственного социального университета. – 2018. – 179 с. – ISBN 978-5-7139-1355-7.

[22] Харари Ф. *Теория графов* // М.: Мир, 1973. – 301 с.

[23] Дистель Р. *Теория графов* // Новосибирск: Издательство института математики, 2002. – 336 с.

[24] Gera R., Hedetniemi S., Larson C. (Eds.) *Graph Theory. Favorite Conjectures and Open Problems – 1*. Berlin: Springer, 2016 – 291 p.

[25] Карпов Д. *Теория графов* // СПб.: Изд-во Санкт-Петербургского отделения Математического института им. В. А. Стеклова РАН, 2017. – 482 с.

[26] Gera R., Hedetniemi S., Larson C. (Eds.) *Graph Theory. Favorite Conjectures and Open Problems – 2*. Berlin: Springer, 2018. – 281 p.

[27] Нидхем М., Ходлер Э. *Графовые алгоритмы. Практическая реализация на платформах Apache Spark и Neo4j*. М.: ДМК Пресс, 2020. – 258 с.

Борис Феликсович МЕЛЬНИКОВ,
профессор Совместного университета МГУ – ППИ
в Шэньчжэне, Китай
(http://szmsubit.ru/),
email$_1$: bormel@smbu.edu.cn,
email$_2$: bf-melnikov@yandex.ru,
mathnet.ru: personid=27967,
elibrary.ru: authorid=15715,
scopus.com: authorId=55954040300,
ORCID: orcidID=0000-0002-6765-6800.

Александра Александровна МЕЛЬНИКОВА,
доцент Димитровградского инженерно-технологического института – филиала Национального исследовательского ядерного университета «МИФИ»
(https://diti-mephi.ru/),
email: super-avahi@yandex.ru,
mathnet.ru: personid=148963,
elibrary.ru: authorid=143351,
scopus.com: authorId=6603567251,
ORCID: orcidID=0000-0002-1658-6857.

On the problems of extracting the root from a given finite language

Boris Melnikov, Aleksandra Melnikova

***Abstract*—The article considers finite languages only.**

Based on the standard definition of the product (concatenation), the non-negative degree of the language is introduced. Root extraction is the inverse operation to it, and it can be defined in several different ways. Despite the simplicity of the formulation of the problem, the authors could not find any description of it in the literature (as well as on the Internet), including even its formulation. Despite this, we believe that all the material of the paper (including the proven theorem) about a possible change in the root-answer it is very easy, and it is puzzling that there are no formulations of such tasks in the monographs known to the authors.

Most of the material in this paper is devoted to the simplest version of the formulation, i.e., the root of the 2-th degree for the 1-letter alphabet; but many of the topics of the paper are generalized to more complex cases. Apparently, for a possible future description of a polynomial solution algorithm, at least one of the described statements of the root extraction tasks first needs to really analyze in detail such a special case, that is: either describe the necessary polynomial algorithm, or, conversely, show that the problem belongs to the class of NP-complete problems.

Thus, in this paper we do not propose a polynomial algorithm for the considered problems; however, the models described here should help in constructing appropriate heuristic algorithms for solving them. A detailed description of the possible further application of such heuristic algorithms is beyond the scope of this article, but several arguments about such a possible application can be carried out already now. Firstly, the possibility of using such algorithms in cryptography and cryptanalysis is quite obvious. Secondly, even if there is a polynomial algorithm for the considered root extraction problem, it seems to be quite complex, and should be called a "intractable" or a "hard problem". Thirdly, the considered problem is easily reduced to an NP-complete problem from the field of graph theory, i.e. the problem of finding a clique; at the same time, the opposite information has not been proven; we also mention the problem of set covering.

***Keywords*—formal languages, iterations of languages, root extraction, algorithms, Boolean.**

References

[1] Salomaa A. Jewels of Formal Language Theory. Computer Science Press, Rockville, Maryland (1981). – 144 p.

[2] Narendran B., Tiwari P. *Polynomial root-finding: analysis and computational investigation of a parallel algorithm* // In: Proceedings of ACM Symposium on Parallelism in Algorithms and Architectures (1992), SPAA'92.

[3] Melnikov B., Korabelshchikova S., Dolgov V. *On the task of extracting the root from the language* // International Journal of Open Information Technologies. – 2019. – Vol. 7. No. 3. – P. 1–6.

[4] Melnikov B. *Semi-lattices of the subsets of potential roots in the problems of the formal languages theory. Part I. Extracting the root from the language* // International Journal of Open Information Technologies. – 2022. – Vol. 10. No. 4. – P. 1–9 (in Russian).

[5] Ginsburg S. *The Mathematical Theory of Context-free Languages*. NY: McGraw-Hill Ed. – 1966. – 245 p.

[6] Aho A., Ullman J. *The Theory of Parsing, Translation, and Compiling. Vol. 1: Parsing*. NJ: Prentice Hall, 1972. – 560 p.

[7] Hopcroft J., Motwani R., Ullman J. *Introduction to Automata Theory, Languages, and Computation*. Massachusetts, Addison-Wesley Publishing Company Reading. – 2006. – 550 p.

[8] Hromkovič J. *Theoretical Computer Science. Introduction to Automata, Computability, Complexity, Algorithmics, Randomization, Communication, and Cryptography*. Berlin, Springer. – 2003. – 321 p.

[9] Garey M., Johnson D. *Computers and Intractability*. USA: W. H. Freeman and Company. – 1979. – 338 p.

[10] Eremeev A., Zaozerskaya L., Kolokolov A. *The problem of covering a set: complexity, algorithms, experimental studies* // Discrete analysis and operations research, ser. 2. – 2000. – Vol. 7. No. 2. – P. 22–46. (In Russian.)

[11] Kameda T., Weiner P. *On the state minimization of nondeterministic finite automata* // IEEE Transactions on Computers. – 1970. – Vol. C-19, No. 7. – P. 617–627.

[12] Jiang T., Ravikumar B. *Minimal NFA problems are hard*. SIAM Journal on Computing. 1993, Vol. 22, No. 6, P. 1117–1141.

[13] Melnikov B. *Subclasses of the context-free languages class (monograph)*. – Moscow, Moscow State University Ed. – 1995. – 174 p. – ISBN 5-211-03448-1. (in Russian).

[14] Melnikov B., Vakhitova A. *Some more on the finite automata* // The Korean Journal of Computational and Applied Mathematics (Journal of Computational and Applied Mathematics). – 1998. – Vol. 5, No. 3. – P. 495–505.

[15] Melnikov B. *A new algorithm of the state-minimization for the nondeterministic finite automata*. The Korean Journal of Computational and Applied Mathematics (Journal of Applied Mathematics and Computing). 1999, Vol. 6, No. 2. – P. 277–290.

[16] Polák L. *Minimalizations of NFA using the universal automaton* // International Journal of Foundation of Computer Science. – 2005. – Vol. 16, No. 5. – P. 999–1010.

[17] Lombardy S., Sakarovitch J. *The universal automaton* // Logic and Automata. Amsterdam University Press. – 2008. – P. 457–504.

[18] Hromkovič J., Petersen H., Schnitger G. *On the limits of the communication complexity technique for proving lower bounds on the size of minimal NFA's*. Theoretical Computer Science. 2009. – No. 410. – P. 2972–2981.

[19] Yo-Sub Han. *State elimination heuristics for short regular expressions*. Fundamenta Informaticae. 2013. – No. 128. – P. 445–462.

[20] Dolgov V., Melnikov B. *On the automatic construction algorithms of Waterloo-like finite automata based on complete automata* // Heuristic algorithms and distributed computing. – 2014. – Vol. 1. No. 4. – P. 24–45 (in Russian).

[21] Melnikov B. *Regular languages and nondeterministic finite automata (monograph)*. – Moscow, Russian Social State University Ed. – 2018. – 179 p. – ISBN 978-5-7139-1355-7. (in Russian).

[22] Harary F. *Graph theory* // Massachusetts: Addison-Wesley Publ. – 1969. – 274 p.

[23] Diestel R. *Graph theory* // Heidelberg: Springer-Verlag, 1997. – 358 p.

[24] Gera R., Hedetniemi S., Larson C. (Eds.) *Graph Theory. Favorite Conjectures and Open Problems – 1*. Berlin:

Springer, 2016 – 291 p.
[25] Karpov D. *Graph theory* // Saint Petersburg: Publishing House of the St. Petersburg Branch Mathematical Institute named after V. A. Steklov of Russian Academy of Sciences, 2017. – 482 p. (in Russian)
[26] Gera R., Hedetniemi S., Larson C. (Eds.) *Graph Theory. Favorite Conjectures and Open Problems – 2*. Berlin: Springer, 2018. – 281 p.
[27] Needham M., Hodler E. *Graph Algorithms. Practical Examples in Apache Spark and Neo4j*. Berlin: O'Reilly Media, Inc. – 2019. – 300 p.

Boris MELNIKOV,
Professor of Shenzhen MSU – BIT University, China
(http://szmsubit.ru/),
email$_1$: bormel@smbu.edu.cn,
email$_2$: bf-melnikov@yandex.ru,
mathnet.ru: personid=27967,
elibrary.ru: authorid=15715,
scopus.com: authorId=55954040300,
ORCID: orcidID=0000-0002-6765-6800.

Aleksandra MELNIKOVA,
Associated Professor of
Dimitrovgrad Engineering and Technology Institute –
Branch of National Research Nuclear University "MEPhI"
(https://diti-mephi.ru/),
email: super-avahi@yandex.ru,
mathnet.ru: personid=148963,
elibrary.ru: authorid=143351,
scopus.com: authorId=6603567251,
ORCID: orcidID=0000-0002-1658-6857.