ВЫЧИСЛИТЕЛЬНЫЕ МЕТОДЫ В ДИСКРЕТНОЙ МАТЕМАТИКЕ

DOI 10.17223/20710410/14/9

УДК 519.7

РЕГУЛЯРНЫЕ ОЦЕНКИ СЛОЖНОСТИ УМНОЖЕНИЯ МНОГОЧЛЕНОВ И УСЕЧЕННОГО ДПФ[1]

И. С. Сергеев

Московский государственный университет им. М. В. Ломоносова, г. Москва, Россия

E-mail: isserg@gmail.com

Строятся схемы для умножения многочленов и усеченного ДПФ (дискретного преобразования Фурье), эффективные с точки зрения сложности и глубины или сложности и объема памяти. Как следствие, умножение многочленов суммарной степени $n-1$, где $n = 2^{n_1} + \ldots + 2^{n_s}$, $n_1 > \ldots > n_s$, над кольцом, в котором обратима двойка, можно выполнить со сложностью $M(n_1) + \ldots + M(n_s) + \mathrm{O}(n)$ арифметических операций в этом кольце и глубиной $\max\limits_i\{D(n_i)\} + \mathrm{O}(\log n)$, где $M(k)$ и $D(k)$ — соответственно сложность и глубина схемы умножения по модулю $x^{2^k}+1$. Усеченное ДПФ порядка n (т. е. ДПФ порядка $2^{\lceil \log_2 n \rceil}$, приведенное к векторам длины n) можно реализовать схемой сложности $1{,}5n\log_2 n + \mathrm{O}(n)$ и объема памяти $n+1$.

Ключевые слова: *арифметические схемы, сложность, глубина, объем памяти, умножение, дискретное преобразование Фурье (ДПФ).*

Введение

В настоящей работе рассматривается вопрос о построении схем для умножения многочленов со сложностью, регулярно (равномерно, гладко) зависящей от размерности входа (т. е. степеней многочленов). Типичной является ситуация, когда известны эффективные (базовые) схемы умножения по модулям $x^{2^k}+1$, но требуется построить схему умножения многочленов произвольной суммарной степени $n-1$ и чтобы при этом она имела сложность, асимптотически так же зависящую от n, как и в случае $n = 2^k$ (асимптотика, естественно, предполагается гладкой функцией). В теоретических работах эта цель обычно достигается видоизменением базового алгоритма умножения, например расширением его на модули вида $x^{s2^k}+1$. Но подобные приемы обычно ухудшают уже второй член в оценке сложности. (Термин «второй член» здесь и далее употребляется неформально.) Интересно, что лучшего результата можно добиться, совсем не вмешиваясь в базовый алгоритм (используя его в качестве «черного ящика»). Для этого достаточно уметь эффективно приводить многочлен по набору модулей $x^{2^i}+1$ и восстанавливать его по известным остаткам от деления на эти же модули. Об этом и пойдет речь далее.

[1]Работа выполнена при финансовой поддержке РФФИ, проекты 11–01–00508 и 11–01–00792–а, и программы фундаментальных исследований Отделения математических наук РАН «Алгебраические и комбинаторные методы математической кибернетики и информационные системы нового поколения» (проект «Задачи оптимального синтеза управляющих систем»).

Вышеуказанной задаче созвучна задача эффективной реализации усеченного ДПФ (в западной литературе TFT — *Truncated Fourier Transform*). Наиболее эффективно вычисляется ДПФ порядка степени двойки, поэтому в работе [1] предложено вместо ДПФ произвольного порядка n использовать n компонент ДПФ порядка $2^{\lceil \log_2 n \rceil}$; соответствующее преобразование в [1] предложено называть усеченным ДПФ.

В работе строятся схемы для решения перечисленных задач, эффективные с точки зрения сложности (потенциально может сохраняться более одного члена из асимптотики сложности базовой схемы) и одновременно глубины или объема требуемой памяти. Постановка задачи о минимизации объема памяти позаимствована из [2].

Дополнительно рассмотрена задача эффективной реализации умножения многочленов при помощи схем умножения по модулям $x^{2\cdot 3^k}+x^{3^k}+1$. Такие схемы предложил использовать для умножения А. Шёнхаге [3]. Метод Шёнхаге обобщается на умножение по модулям $x^{(p-1)\cdot p^k} + x^{(p-2)\cdot p^k} + \ldots + x^{p^k} + 1$, где p — простое число [4]. Однако при $p \neq 3$ это обобщение, по всей видимости, практического значения пока не имеет (см. также [5, 6]). Эти схемы обычно применяются тогда, когда двойка необратима в кольце коэффициентов, зато обратима тройка (популярный пример — конечные поля характеристики 2). В этом случае, как показано ниже, для умножения многочленов суммарной степени $n-1$ можно построить схемы с регулярной оценкой сложности для $n \in \bigcup\limits_{k\in\mathbb{N}} [2\cdot 3^k,\ 3^{k+1})$ и с ухудшением асимптотики сложности в не более чем $4/3$ раза для прочих n при условии, что базовые схемы не модифицируются.

Изложим существо вопроса более подробно, попутно вводя необходимые понятия.

Пусть $\mathbf{K}$ — коммутативное (и ассоциативное) кольцо с единицей. Элемент $\zeta \in \mathbf{K}$ называется *примитивным корнем* степени N из единицы, если $\zeta^N = 1$ и при любом простом $p|N$ элемент $\zeta^{N/p} - 1$ не является делителем нуля в $\mathbf{K}$.

Дискретным преобразованием Фурье порядка N называется $(\mathbf{K}^N \to \mathbf{K}^N)$-преобразование

$$\text{ДПФ}_{N,\zeta}(\gamma_0, \ldots, \gamma_{N-1}) = (\gamma_0^*, \ldots, \gamma_{N-1}^*), \qquad \gamma_j^* = \sum_{i=0}^{N-1} \gamma_i \zeta^{ij},$$

где ζ — примитивный корень степени N. Если элемент $N = 1 + \ldots + 1 \in \mathbf{K}$ обратим, то существует обратное к ДПФ преобразование (называемое *обратным ДПФ*), удовлетворяющее соотношению $\text{ДПФ}_{N,\zeta}^{-1} = (1/N)\text{ДПФ}_{N,\zeta^{-1}}$.

Удобно иметь в виду полиномиальную интерпретацию ДПФ как изоморфизма колец, отождествляя аргументы преобразования с вектором коэффициентов многочлена:

$$\text{ДПФ}_{N,\zeta} : \mathbf{K}[x]/(x^N - 1) \to \mathbf{K}^N, \quad \text{ДПФ}_{N,\zeta}(\Gamma(x)) = \left(\Gamma(\zeta^0), \ldots, \Gamma(\zeta^{N-1})\right).$$

Подробнее о ДПФ см., например, в [6].

Для анализа эффективности обсуждаемых далее алгоритмов удобно использовать стандартную вычислительную модель *схемы из функциональных элементов* [7, 8] (далее просто *схемы*). А именно, рассматриваются схемы над арифметическим базисом $\{x \pm y,\ xy\} \cup \{ax : a \in \mathbf{K}\}$ (*арифметические схемы*). Для программистов ближе понятие *неветвящейся программы*: по существу, неветвящаяся программа (далее просто *программа*) — это схема, для которой фиксирована последовательность выполнения операций (срабатывания элементов схемы). Таким образом, одной схеме могут соответствовать несколько программ.

Стандартным образом определяются несколько мер сложности схем (программ). Собственно *сложность* — число функциональных элементов в схеме (программе). *Глубина* схемы — максимальное число элементов в ориентированной цепи, соединяющей

вход и выход схемы. *Объем памяти* программы — максимальное по всем итерациям число промежуточных данных (включая все уже вычисленные выходы), используемых в последующих итерациях. Несколько искусственно объем памяти схемы можно определить как минимальный объем памяти по всем программам, соответствующим данной схеме.

Содержательно сложность отвечает времени выполнения программы на однопроцессорной машине или площади микросхемы, реализующей схему; глубина отвечает времени срабатывания микросхемы или числу параллельных шагов, выполняемых многопроцессорной машиной; объем памяти в определении соответствует объему памяти, отводимому для хранения переменных при реализации программы на ЭВМ.

При анализе объема памяти процесс вычислений удобно представлять следующим образом. Промежуточные данные хранятся в ячейках памяти (емкость ячейки равна одному элементу кольца $\mathbf{K}$). В начале работы ячейки содержат входные данные. Элементарные (базисные) операции выполняются последовательно. Каждая операция использует данные, находящиеся в памяти в момент ее выполнения. Результат выполнения операции сохраняется в некоторой ячейке.

Наиболее эффективно реализуется ДПФ порядка степени двойки. Как следствие, быстрые алгоритмы эффективно умножают многочлены суммарной степени $n-1$, если $n = 2^k$, в предположении, что кольцо коэффициентов или подходящее его расширение допускает ДПФ порядка степени двойки.

Как известно, методом Кули — Тьюки [9] ДПФ порядка 2^k может быть реализовано схемой из $k2^k$ элементов сложения-вычитания, $(k-2)2^{k-1}+1$ элементов умножения на степени примитивного корня (степени 2^k), глубины $2k-1$ (если учитывать только аддитивные операции, то глубина равна k). Эта схема может быть перестроена так, чтобы с той же сложностью иметь объем памяти 2^k+1. Добавив $k2^{k-1}$ элементов умножения на 2, объем памяти можно сократить до 2^k; здесь имеется в виду, что основное преобразование схемы, ДПФ порядка 2 $(x, y) \to (x+y, x-y)$, можно вычислить со сложностью 2 и объемом памяти 3 в цепочке $(x, y) \to (x+y, x, y) \to (x+y, x-y)$ или со сложностью 3 и объемом памяти 2 в цепочке $(x, y) \to (x + y, y) \to (x + y, 2y) \to (x + y, x - y)$.

Обратное ДПФ можно реализовать так же, как и прямое, заменив ζ на ζ^{-1} и выполнив в конце 2^k умножений на константу $2^{-k} \in \mathbf{K}$. Впрочем, часть этих умножений можно совместить с внутренними умножениями на степени примитивного корня.

Если $n \neq 2^k$, то «по умолчанию» можно использовать алгоритм умножения многочленов суммарной степени $2^{\lceil \log_2 n \rceil} - 1$. Но при этом сложность алгоритма умножения с ростом n растет нерегулярно, возникают примерно двукратные скачки при переходе через степени двойки. Известны приемы, позволяющие сгладить функцию сложности так, чтобы при любом n она асимптотически так же выражалась через n, как и в случае $n = 2^k$. Например, ДПФ порядка степеней двойки можно заменить на ДПФ порядка $s2^l$, где $s \ll 2^l$ [10], или чуть более общими многократными ДПФ [11]. Указанные приемы, однако, как правило, ухудшают второй по порядку роста член функции сложности и переносят скачки в него.

Другой подход предложил Ван дер Хувен в [1, 12], где реализуется усеченное ДПФ, вычисляющее значение ДПФ на некотором подмножестве точек из множества, соответствующего ДПФ порядка ближайшей сверху степени двойки. *Усеченное ДПФ* (*УДПФ*) *порядка* n определяется как набор из n некоторых компонент вектора $\text{ДПФ}_{2^{\Lambda(n)},\zeta}(\gamma_0, \ldots, \gamma_{n-1}, 0, \ldots, 0)$, где $\Lambda(n) = \lceil \log_2 n \rceil$.

Ван дер Хувен [1] предложил в УДПФ включать компоненты — значения в точках $\zeta^{\rho(i)}$, $i = 0, \ldots, n-1$, где $\rho(i)$ — число, двоичная запись которого получается из двоичной записи числа i дополнением нулями до $\Lambda(n)$ разрядов и последующим обращением.

Схема прямого УДПФ в методе [1] получается из схемы ДПФ порядка $2^{\Lambda(n)}$ удалением «ненужных» элементов. Схема обратного УДПФ устроена несколько сложнее. Для обеих схем получена оценка сложности $n\Lambda(n) + 2^{\Lambda(n)}$ аддитивных операций и $\lceil (n\Lambda(n) + 2^{\Lambda(n)})/2 \rceil$ умножений на степени примитивного корня (умножения на степени двойки в расчете игнорируются, точнее, допускаются операции вида $x \pm 2^s y$). Глубина схемы прямого УДПФ равна $\Lambda(n)$; глубина схемы обратного УДПФ несколько больше, но также $\mathrm{O}(\log n)$. Объем памяти этой схемы (точнее, схемы, перестроенной аналогично упомянутой выше схеме обычного ДПФ), как отмечено в [2], равен $2^{\Lambda(n)}$. В [2] приведена конструкция схемы с объемом памяти $n + \mathrm{O}(1)$, однако большей асимптотической сложности, хотя по порядку и той же самой, $\mathrm{O}(n \log n)$.

С позиции задачи умножения многочленов близкие к использованию УДПФ идеи ранее высказывались в [13], где предлагалось выполнять умножение по двум модулям, допускающим быстрый алгоритм умножения, и окончательно восстанавливать произведение, опираясь на китайскую теорему об остатках, и позже в [5], где предлагалось использовать несколько модулей. На языке ДПФ это означает выбор УДПФ порядка l, где число l имеет малый (двоичный) вес.

В работе [14] предложено в качестве точек УДПФ порядка n выбирать корни многочленов $x^{2^i} + 1$ суммарной степени n.

Для дальнейшего анализа удобно ввести понятие *нечетного ДПФ* (*НДПФ*) *порядка* N:

$$\text{НДПФ}_{N,\zeta} : \mathbf{K}[x]/(x^N + 1) \to \mathbf{K}^N, \quad \text{НДПФ}_{N,\zeta}(\Gamma(x)) = \left(\Gamma(\zeta^1), \Gamma(\zeta^3), \ldots, \Gamma(\zeta^{2N-1})\right),$$

где ζ — примитивный корень степени $2N$. Другими словами, компонентами НДПФ порядка N являются компоненты ДПФ порядка $2N$, отличные от компонент ДПФ порядка N (если иметь в виду полиномиальную интерпретацию ДПФ). Простой способ реализации $\text{НДПФ}_{N,\zeta}$ состоит в композиции замены переменной $x \to \zeta x$ и $\text{ДПФ}_{N,\zeta^2}$. Соответственно обратное НДПФ можно реализовать как композицию $\text{ДПФ}^{-1}_{N,\zeta^2}$ и замены переменной $x \to x/\zeta$.

Пусть $n = 2^{n_1} + \ldots + 2^{n_s}$, где $n_1 > \ldots > n_s$. Схема УДПФ [14] строится из схем НДПФ порядков $2^{n_1}, \ldots, 2^{n_s}$ плюс дополнительно используется $2^{\Lambda(n)} + n$ аддитивных операций и n умножений. В схеме обратного УДПФ, помимо схем обратных НДПФ порядков $2^{n_1}, \ldots, 2^{n_s}$, используется $3 \cdot 2^{\Lambda(n)} + n$ аддитивных операций и n умножений. Указанные оценки сложности рассчитаны в [14] для чуть более общей задачи и могут быть уменьшены. Глубина и объем памяти схем [14] оцениваются приблизительно так же, как и для [1].

Далее получены несколько более точные оценки сложности данного УДПФ вместе с оценками глубины и показано, что УДПФ порядка n можно реализовать схемой с объемом памяти n (или $n + 1$) и дополнительной сложностью $\mathrm{O}(n)$.

Вопрос об оптимизации объема памяти естественно рассмотреть и в задаче умножения многочленов. В работе [2] на основе эффективной схемы для УДПФ построена схема умножения многочленов степени $n/2 - 1$ со сложностью $\mathrm{O}(n \log n)$ и объемом памяти $2n + \mathrm{O}(1)$, причем n «ячеек» памяти, отводимых под хранение коэффициентов исходных многочленов, не модифицируются.

Из полученных далее оценок вытекает, в частности, что для умножения многочленов суммарной степени $n - 1$ можно построить схему сложности $\mathrm{O}(n \log n)$ и объема

памяти $2n$. Запрет на переписывание коэффициентов исходных многочленов при этом не накладывается. Если иметь в виду, что фактически строится схема для умножения по модулю некоторого многочлена степени n, то объем памяти такой схемы не может быть далее понижен.

Результаты для задач реализации УДПФ и умножения многочленов формулируются в п. 1.1 и доказываются в п. 1.2. В п. 2 схемы для умножения многочленов строятся из схем умножения по модулям $x^{2\cdot 3^k} + x^{3^k} + 1$.

1. УДПФ и двоичный метод умножения

1.1. Основные результаты

Пусть в нашем распоряжении есть схемы для НДПФ порядка 2^k сложности $\Phi(k) = \Phi_A(k) + \Phi_2(k) + \Phi_C(k)$, глубины $d_\Phi(k)$ и объема памяти $v_\Phi(k)$. Здесь $\Phi_A(k)$ — число аддитивных элементов; $\Phi_2(k)$ — число умножений на степени двойки; $\Phi_C(k)$ — число прочих скалярных умножений. Аналогичные обозначения со штрихами введем для параметров схем обратных НДПФ.

Зафиксируем обозначение $n = 2^{n_1} + \ldots + 2^{n_s}$, где $n_1 > \ldots > n_s$.

Теорема 1. Усеченное ДПФ порядка n можно реализовать схемой

а) из $2n + \sum\limits_i \Phi_A(n_i)$ аддитивных операций, $\sum\limits_i \Phi_2(n_i)$ и $\sum\limits_i \Phi_C(n_i)$ умножений на степени двойки и прочие константы соответственно, и глубины $n_1 + \max\limits_i\{d_\Phi(n_i) - n_i\} + 1$;

б) из $4n - 2^{n_1+1} + \sum\limits_i \Phi_A(n_i)$ аддитивных операций, $4n - 3\cdot 2^{n_1} + \sum\limits_i \Phi_2(n_i)$ и $\sum\limits_i \Phi_C(n_i)$ умножений на степени двойки и прочие константы соответственно, и объема памяти $n + \max\limits_i\{v_\Phi(n_i) - 2^{n_i}\}$.

Обратное усеченное ДПФ порядка n можно реализовать схемой

а) из $4n - 3\cdot 2^{n_1} + \sum\limits_i \Phi'_A(n_i)$ аддитивных операций, $2n - 2^{n_1+1} + \sum\limits_i \Phi'_2(n_i)$ и $\sum\limits_i \Phi'_C(n_i)$ умножений на степени двойки и прочие константы соответственно, и глубины $n_1 - n_s + {}$ $+ 2s - 1 + \max\limits_i\{d'_\Phi(n_i)\}$;

б) из $4n - 2^{n_1+1} + \sum\limits_i \Phi'_A(n_i)$ аддитивных операций, $2n - 2^{n_1+1} + \sum\limits_i \Phi'_2(n_i)$ и $\sum\limits_i \Phi'_C(n_i)$ умножений на степени двойки и прочие константы соответственно, и объема памяти $n + \max\limits_i\{v'_\Phi(n_i) - 2^{n_i}\}$.

Теорема вытекает непосредственно из доказываемых ниже лемм 2 и 3.

Конкретные оценки можно получить при подстановке параметров схем НДПФ, указанных во введении. Например, $\Phi_A(k) = \Phi'_A(k) = k2^k$, $\Phi_C(k) = \Phi'_C(k) = k2^{k-1}$, $\Phi_2(k) = 0$, $\Phi'_2(k) = 2^k$ и либо $d_\Phi(k) = d'_\Phi(k) = 2k$, либо $v_\Phi(k) = v'_\Phi(k) = 2^k + 1$.

Теперь пусть имеются схемы для умножения многочленов по модулям $x^{2^k}+1$ с коэффициентами над кольцом, в котором обратим элемент 2. Сложность, глубину и объем памяти таких схем будем обозначать через $M(k) = M_A(k) + M_2(k) + M_C(k) + M_N(k)$, $d_M(k)$ и $v_M(k)$, где $M_N(k)$ обозначает число нескалярных умножений в схеме (остальные обозначения аналогичны введенным выше для схем НДПФ). Из лемм 2 и 3 также следует

Теорема 2. Для умножения многочленов суммарной степени $n-1$ можно построить схему

а) из $6n-3\cdot 2^{n_1}+\sum\limits_i M_A(n_i)$ аддитивных операций, $2n-2^{n_1+1}+\sum\limits_i M_2(n_i)$, $\sum\limits_i M_C(n_i)$ и $\sum\limits_i M_N(n_i)$ умножений на степени двойки, прочие константы и нескалярных умножений соответственно, и глубины $\max\limits_i\{d_M(n_i)-n_i\}+2n_1-n_s+2s$;

б) из $12n-6\cdot 2^{n_1}+\sum\limits_i M_A(n_i)$ аддитивных операций, $10n-8\cdot 2^{n_1}+\sum\limits_i M_2(n_i)$, $\sum\limits_i M_C(n_i)$ и $\sum\limits_i M_N(n_i)$ умножений на степени двойки, прочие константы и нескалярных умножений соответственно, и объема памяти $2n+\max\limits_i\{v_M(n_i)-2^{n_i+1}\}$.

Учитывая, что $s\leqslant n_1+1$, получаем

Следствие 1. Пусть $M(k)=f(2^k)$ и для любых $x,y\geqslant 1$ справедливо $f(x+y)\geqslant \geqslant f(x)+f(y)$. Пусть также $d_M(k)-k\leqslant d_M(l)-l$ и $v_M(k)-2^{k+1}\leqslant v_M(l)-2^{l+1}$ при любых $k\leqslant l$. Тогда для умножения многочленов суммарной степени $n-1$ можно построить схему

а) сложности $f(n)+8n-5\cdot 2^{n_1}$ и глубины $d_M(n_1)+3n_1+2$;

б) сложности $f(n)+22n-14\cdot 2^{n_1}$ и объема памяти $2n+v_M(n_1)-2^{n_1+1}$.

Выбирая ближайшее сверху к n число n', кратное $2^{n_1-\alpha(n)}$, где $\alpha(n)$ — медленно растущая натуральная функция, и переходя к схеме умножения многочленов суммарной степени не выше $n'-1$, получаем еще одно следствие из теоремы 2, п. *а* (используя неравенства $n_1-n_s\leqslant\alpha(n)$, $s\leqslant\alpha(n)+1$):

Следствие 2. В условиях следствия 1 дополнительно предположим, что $f(x)/x\to\infty$ при $x\to\infty$ и $f(x)=x^{\mathrm{O}(1)}$. Тогда для умножения многочленов суммарной степени $n-1$ можно построить схему сложности не более $(1+\mathrm{o}(1))f(n)$ и глубины не более $d_M(\lfloor\log_2(n+o(n))\rfloor)+\mathrm{o}(\log n)$.

Заметим, что предположения в формулировках следствий являются естественными. Во-первых, исходим из того, что схемы умножения по модулям $x^{2^k}+1$ устроены единообразно. Во-вторых, предполагаем, что частично определенная функция сложности схем умножения $M(k)$, $k\in\{2^i : i\in\mathbb{N}\}$, может быть доопределена до функции $f(x)$, удовлетворяющей условию суперлинейности $f(x+y)\geqslant f(x)+f(y)$ (следствие 1) и имеющей нелинейный рост (следствие 2). Первое из условий на $f(x)$ фактически следует из второго, а второе опирается на широко распространенное предположение о нелинейности функции сложности умножения. Если какое-то из условий все же не может быть удовлетворено, то вместо $f(x)$ можно взять подходящую функцию $f_>(x)\geqslant f(x)$, удовлетворяющую обоим условиям.

Конкретные оценки можно получить при подстановке известных параметров схем умножения по модулю $x^{2^k}+1$ — они приводятся, например, в [5, 6, 14, 15].

Стандартным образом схему умножения по модулю $x^{2^k}+1$ можно построить из двух схем НДПФ, схемы обратного НДПФ порядка 2^k (если ДПФ порядка 2^{k+1} существует в рассматриваемом кольце) и 2^k нескалярных умножений, выполняемых на одном уровне. Подставляя известные параметры этих схем, в частности для схемы из п. *б* теоремы 2 получаем оценку сложности $\mathrm{O}(n\log n)$ при объеме памяти $2n$.

1.2. Вспомогательные утверждения

Пусть $a(x)=\sum\limits_{l=0}^{n-1}a_lx^l$. Положим формально $a_l=0$ при $l\geqslant n$. Введем обозначения $a_{k,l}$ и $b_{k,l}$:

$$a(x)\bmod(x^{2^k}-1)=\sum_{l=0}^{2^k-1}a_{k,l}x^l,\quad a(x)\bmod(x^{2^k}+1)=\sum_{l=0}^{2^k-1}b_{k,l}x^l.$$

Очевидно, что коэффициенты $a_{k,l}$ и $b_{k,l}$ являются соответственно знакопостоянными и знакопеременными суммами коэффициентов a_i с шагом 2^k (по индексам):

$$a_{k,l} = a_{k+1,l} + a_{k+1,2^k+l} = \sum_{j2^k<n-l} a_{j2^k+l}, \quad b_{k,l} = a_{k+1,l} - a_{k+1,2^k+l} = \sum_{j2^k<n-l} (-1)^j a_{j2^k+l}.$$

Следующая лемма используется в первую очередь при построении схем, эффективных с точки зрения памяти.

Лемма 1. Пусть $n_{i+1} < k \leqslant n_i$. Справедливы следующие формулы:

$$a_{k,l} = \sum_{j_i=0}^{2^{n_i-k}-1} \left(b_{n_i,\, j_i 2^k+l} + 2 \sum_{j_{i-1}=0}^{2^{n_{i-1}-n_i-1}-1} \left(b_{n_{i-1},\, (2j_{i-1}+1)2^{n_i}+j_i 2^k+l} + \right.\right.$$
$$+2 \sum_{j_{i-2}=0}^{2^{n_{i-2}-n_{i-1}-1}-1} \left(b_{n_{i-2},\, (2j_{i-2}+1)2^{n_{i-1}}+(2j_{i-1}+1)2^{n_i}+j_i 2^k+l} + \dots \right.$$
$$\left.\left.\left. \dots + 2 \sum_{j_1=0}^{2^{n_1-n_2-1}-1} b_{n_1,\, (2j_1+1)2^{n_2}+\dots+(2j_{i-1}+1)2^{n_i}+j_i 2^k+l} \dots \right) \right) \right) + 2^i a_{2^{n_1}+\dots+2^{n_i}+l};$$

$$b_{n_i,l} = \sum_{j=0}^{1} (-1)^j \left(\sum_{j_{i-1}=0}^{2^{n_{i-1}-n_i-1}-1} \left(b_{n_{i-1},\, (2j_{i-1}+j)2^{n_i}+l} + \right.\right.$$
$$+2 \sum_{j_{i-2}=0}^{2^{n_{i-2}-n_{i-1}-1}-1} \left(b_{n_{i-2},\, (2j_{i-2}+1)2^{n_{i-1}}+(2j_{i-1}+j)2^{n_i}+l} + \dots \right.$$
$$\left.\left.\left. \dots + 2 \sum_{j_1=0}^{2^{n_1-n_2-1}-1} b_{n_1,\, (2j_1+1)2^{n_2}+\dots+(2j_{i-2}+1)2^{n_{i-1}}+(2j_{i-1}+j)2^{n_i}+l} \dots \right) \right) \right) +$$
$$+2^{i-1} \left(a_{2^{n_1}+\dots+2^{n_{i-1}}+l} - a_{2^{n_1}+\dots+2^{n_i}+l} \right).$$

Доказательство. Первая из формул получается рекурсивным применением простых соотношений

$$a_{k,l} = \begin{cases} a_{k+1,l} + a_{k+1,2^k+l}, & k \notin \{n_i : i = 1, \dots, s\}, \\ b_{k,l} + 2a_{k+1,2^k+l}, & k \in \{n_i : i = 1, \dots, s\}, \end{cases}$$

отталкиваясь от $a_{n_1+1,l} = a_l$ и учитывая, что $a_l = 0$ при $l \geqslant n$. Вторая формула получается из первой ввиду $b_{n_i,l} = a_{n_i+1,l} - a_{n_i+1,2^{n_i}+l}$. ■

Лемма 2. Пусть $m \leqslant n$. Тогда приведение многочлена степени $m-1$ по модулям $x^{2^{n_i}}+1$, $i = 1, \dots, s$, может быть выполнено схемой

а) из $2(m-1)$ аддитивных элементов и глубины $n_1 - n_s + 1$, причем коэффициенты остатка от деления на $x^{2^{n_i}}+1$ вычисляются на глубине $n_1 - n_i + 1$;

б) из $2(2n - 2^{n_1})$ аддитивных элементов, $4n - 3 \cdot 2^{n_1}$ умножений на степени двойки и объема памяти n.

Доказательство. Пункт *а* доказывает простая конструкция [14].

Несложно видеть, что все суммы $a_{k,l}$, где $k = n_s+1, \dots, n_1$, можно вычислить одной схемой сложности не выше $m-1$, в которой на глубине не более l вычисляются $a_{k,l}$ при $k = n_1 + 1 - l$.

Искомые коэффициенты $b_{n_i,l}$ получаются присоединением к построенной схеме элементов вычитания, расположенных на глубине $n_1 - n_i + 1$ в количестве

$$\begin{cases} 0, & m \leqslant 2^{n_i}, \\ m - 2^{n_i}, & 2^{n_i} \leqslant m \leqslant 2^{n_i+1}, \\ 2^{n_i}, & m \geqslant 2^{n_i+1}. \end{cases}$$

Поскольку $2^{n_i} > 2^{n_{i+1}} + \ldots + 2^{n_s}$, для числа вычитаний (при подходящем i) получаем оценку

$$m - 2^{n_i} + 2^{n_{i+1}} + \ldots + 2^{n_s} \leqslant m - 1,$$

откуда следует оценка $2(m-1)$ для сложности всей схемы.

Доказательство п. *б* достаточно провести для случая $m = n$. Перестроим схему из предыдущего пункта так, чтобы в ней явно вычислялись только те суммы $a_{k,l}$, для которых «достаточно памяти».

Обозначим $L_i = 2^{n_i} + \ldots + 2^{n_s}$. При любом $k = n_1, \ldots, n_s + 1$ явно вычисляем суммы $a_{k,0}, \ldots, a_{k,L_i-1}$, где $n_{i-1} \geqslant k > n_i$, учитывая, что $n - L_i$ прочих «ячеек памяти» отводится под хранение коэффициентов $b_{n_j,l}$, где $j < i$.

Все вычисление удобно разбить на этапы, нумеруя их числами от n_1 до n_s в порядке убывания. На этапе k вычисляются суммы $a_{k,0}, \ldots, a_{k,L_i-1}$ (где $n_{i-1} \geqslant k > n_i$) и, если $k \in \{n_i : i = 1, \ldots, s\}$, то вычисляются также коэффициенты $b_{k,0}, \ldots, b_{k,2^k-1}$.

Если сумма $a_{k,l}$ не вычисляется явно, то вместо нее используется правая часть формулы (обозначим ее $\varphi_{k,l}$) из леммы 1, выражающая ее через уже вычисленные коэффициенты $b_{n_j,t}$, где $n_j \geqslant k$. Заметим, что поскольку $l \geqslant L_{i+1}$ при $n_i \geqslant k > n_{i+1}$, то последнее слагаемое $a_{2^{n_1}+\ldots+2^{n_i}+l}$ в $\varphi_{k,l}$ равно нулю.

Обозначим через ρ_k число переменных в формуле $\varphi_{k,l}$ (это число не зависит от l, что следует из вида формулы). Тогда прибавление (вычитание) $\varphi_{k,l}$ выполняется за ρ_k аддитивных операций и i умножений на степени двойки без дополнительной памяти, где $n_i \geqslant k > n_{i+1}$.

Соответствующий способ проиллюстрируем на примере. Пусть требуется выполнить преобразование $a \to a + (b + 2c + 4d)$. Вычисления проведем в следующем порядке: a, $a + b$, $2^{-1}(a + b)$, $c + 2^{-1}(a + b)$, $2^{-1}(c + 2^{-1}(a + b))$, $d + 2^{-1}(c + 2^{-1}(a + b))$, $2^2(d + 2^{-1}(c + 2^{-1}(a + b))) = a + b + 2c + 4d$.

Оценим сложность схемы. Рассмотрим этап с номером k, где $n_{i-1} > k > n_i$. Доступны коэффициенты $a_{k+1,0}, \ldots, a_{k+1,L_i-1}$, остальные $a_{k+1,l}$ выражаются формулами $\varphi_{k+1,l}$. Заметим, что $L_i < 2^{n_i+1} \leqslant 2^k$. Тогда каждый из коэффициентов $a_{k,l}$, где $l < L_i$, вычисляется как $a_{k+1,l} + \varphi_{k+1,2^k+l}$ со сложностью ρ_{k+1} аддитивных операций и $i-1$ умножений на степени двойки. Сложность этапа оценивается как $L_i\rho_{k+1}$ аддитивных операций и $(i-1)L_i$ умножений на степени двойки.

Рассмотрим этап с номером $k = n_i$. Доступны коэффициенты $a_{k+1,0}, \ldots, a_{k+1,L_i-1}$. При этом $L_i = 2^k + L_{i+1}$. Сначала вычислим все $b_{k,l}$ по формулам $a_{k+1,l} - a_{k+1,2^k+l}$ при $l < L_{i+1}$ и $a_{k+1,l} - \varphi_{k+1,2^k+l}$ при остальных l. При этом коэффициентами $b_{k,l}$ «перезаписываются» $a_{k+1,l}$, $l \geqslant L_{i+1}$. Затем вычислим $a_{k,l}$ для $l < L_{i+1}$ по формулам $2a_{k+1,l} - b_{k,l}$. Сложность этапа оценивается как $2L_{i+1} + (2^k - L_{i+1})\rho_{k+1}$ аддитивных операций и $(i-1)2^k - (i-2)L_{i+1} = (i-1)L_i - (2i-3)L_{i+1}$ умножений на степени двойки.

Найдем ρ_k. Из вида формулы леммы 1 непосредственно следует, что

$$\rho_{n_i} = 2^{n_1-n_i-(i-1)} + 2^{n_2-n_i-(i-2)} + \ldots + 2^{n_{i-1}-n_i-1} + 1, \tag{1}$$

$\rho_k = 2^{n_i-k}\rho_{n_i}$ при $n_{i+1} < k < n_i$.

Оценим суммарную аддитивную сложность вычислений. Сумма сложностей этапов $n_i - 1, \ldots, n_{i+1}$ не превосходит

$$C_i = L_{i+1}\rho_{n_i}(1 + 2 + \ldots + 2^{n_i-n_{i+1}-2}) + 2L_{i+2} + (2^{n_{i+1}} - L_{i+2})2^{n_i-n_{i+1}-1}\rho_{n_i} \leqslant$$
$$\leqslant \rho_{n_i}2^{n_i-n_{i+1}-1}(L_{i+1} - L_{i+2} + 2^{n_{i+1}}) - \rho_{n_i}L_{i+1} + 2L_{i+2} = \rho_{n_i}(2^{n_i} - L_{i+1}) + 2L_{i+2}.$$

При $i \geqslant 1$ последнее выражение не превосходит $2^{n_i}\rho_{n_i}$, если учесть, что $L_{i+1} > 2L_{i+2}$ и $\rho_{n_i} \geqslant 1$.

Для сложности всех этапов, кроме этапа n_1, используя (1), получаем оценку

$$C_1 + \ldots + C_{s-1} = \sum_{i=1}^{s-1} 2^{n_i}\rho_{n_i} = \sum_{i=1}^{s-1}(2^{n_1-(i-1)} + 2^{n_2-(i-2)} + \ldots + 2^{n_i}) <$$
$$< (2^{n_1} + 2^{n_1-1} + \ldots) + (2^{n_2} + 2^{n_2-1} + \ldots) + \ldots + (2^{n_s} + 2^{n_s-1} + \ldots) < 2n.$$

Окончательно, оценивая сложность этапа n_1 как $2L_2 = 2(n - 2^{n_1})$, получаем утверждение п. *б* в части аддитивной сложности.

Число умножений на степени двойки на этапах $n_i - 1, \ldots, n_{i+1}$ оценим грубо как

$$D_i = (n_i - n_{i+1} - 1)iL_{i+1} + iL_{i+1} - (2i-1)L_{i+2} \leqslant$$
$$\leqslant i(n_i - n_{i+1})L_{i+1} \leqslant i2^{n_i-n_{i+1}-1}2^{n_{i+1}+1} = i2^{n_i}.$$

Тогда для суммарного числа умножений на всех этапах, кроме этапа n_1, имеем оценку

$$D_1 + \ldots + D_{s-1} = 2^{n_1} + 2 \cdot 2^{n_2} + 3 \cdot 2^{n_3} + \ldots + (s-1) \cdot 2^{n_{s-1}} =$$
$$= n + (n - 2^{n_1}) + (n - 2^{n_1} - 2^{n_2}) + \ldots <$$
$$< n + (n - 2^{n_1}) + (1/2)\,(n - 2^{n_1}) + (1/2)^2\,(n - 2^{n_1}) + \ldots = n + 2(n - 2^{n_1}).$$

Добавляя $n - 2^{n_1}$ умножений на 2 на этапе n_1, получаем итоговую оценку. ■

Лемма 3. Восстановление многочлена степени $n-1$ по заданным остаткам по модулям $x^{2^{n_i}} + 1$, $i = 1, \ldots, s$, может быть выполнено схемой

а) из $4n - 3 \cdot 2^{n_1}$ аддитивных элементов, $2(n - 2^{n_1})$ умножений на степени двойки и глубины не более $n_1 - n_s + 2s - 1$;

б) из $2(2n - 2^{n_1})$ аддитивных элементов, $2(n - 2^{n_1})$ делений на 2 и объема памяти n.

Доказательство. Воспользуемся леммой 1, чтобы выразить разности

$$h_{i,l} = a_{2^{n_1}+\ldots+2^{n_{i-1}}+l} - a_{2^{n_1}+\ldots+2^{n_i}+l}, \tag{2}$$

где $i = 1, \ldots, s-1$ и $l = 0, \ldots, L_i - 1$, через коэффициенты $b_{n_j,t}$. Заметим, что вычитаемый коэффициент равен нулю при $l \geqslant L_{i+1}$.

Для построения схемы из п. *а* вычисляем вспомогательные величины $c_{i,l}$ и $d_{i,l}$, определяемые равенствами

$$d_{1,l} = b_{n_1,l}, \quad c_{i,l} = \sum_{j=0}^{2^{n_{i-1}-n_i-1}-1} d_{i-1,\,j2^{n_i+1}+l}, \quad d_{i,l} = b_{n_i,l} + 2c_{i,2^{n_i}+l},$$

где $i \geqslant 2$. Затем находим искомые коэффициенты $h_{i,l}$ по формулам $h_{i,l} = 2^{1-i}(b_{n_i,l} - c_{i,l} + c_{i,2^{n_i}+l})$, а в случае $i = 1$ просто $h_{1,l} = b_{n_1,l}$.

Сложность вычислений, заключенная в формулах для $c_{i,l}$, $i = 2, \ldots, s$ и $l = 0, \ldots, 2^{n_i+1} - 1$, оценивается как

$$\sum_{i=2}^{s} 2^{n_i+1}(2^{n_{i-1}-n_i-1} - 1) = \sum_{i=2}^{s}(2^{n_{i-1}} - 2^{n_i+1}) < n - 2(n - 2^{n_1}).$$

Сложность, заключенная в формулах для $d_{i,l}$, $i = 2, \ldots, s-1$ и $l = 0, \ldots, 2^{n_i} - 1$, оценивается как $2^{n_2}+\ldots+2^{n_{s-1}} < n-2^{n_1}$ аддитивных операций и столько же умножений на 2. Для завершения вычисления $h_{i,l}$ нужно выполнить еще $2(2^{n_2}+\ldots+2^{n_s}) = 2(n-2^{n_1})$ аддитивных операций и $n - 2^{n_1}$ умножений на степени двойки.

Без труда проверяется, что $c_{i,l}$ при этом вычисляется на глубине $n_1 - n_i + i - 3$ и, следовательно, $h_{i,l}$ — на глубине $n_1 - n_i + i$.

По набору $h_{i,l}$ несложно восстанавливаются коэффициенты многочлена $a(x)$. Коэффициенты $a_{2^{n_1}+\ldots+2^{n_{i-1}}+l}$ просто совпадают с $h_{i,l}$ при $l \geqslant L_{i+1}$. В частности, в случае $i = s$ известны все коэффициенты. Это позволяет последовательно в порядке убывания i определить недостающие коэффициенты $a_{2^{n_1}+\ldots+2^{n_{i-1}}+l}$, $l = 0, \ldots, L_{i+1} - 1$, прямо по формулам (2) с общей сложностью

$$L_s + L_{s-1} + \ldots + L_2 = 2^{n_2} + 2 \cdot 2^{n_3} + 3 \cdot 2^{n_4} + \ldots < 2(n - 2^{n_1}).$$

Глубина этих вычислений, очевидно, не превосходит $s - 1$.

Складывая все оценки, получаем утверждение п. *а*.

Для доказательства п. *б* построим схему, последовательно в порядке убывания i преобразующую каждый коэффициент $b_{n_i,l}$ по формуле леммы 1 в соответствующую разность $h_{i,l}$.

Несложно видеть, что преобразование каждого коэффициента $b_{n_i,l}$ выполняется за σ_i аддитивных операций и $i - 1$ делений на 2, где σ_i — число коэффициентов $b_{n_j,t}$ в правой части второй формулы леммы 1. Непосредственно проверяется, что

$$\sigma_i = 2^{n_1-n_i-(i-2)} + 2^{n_2-n_i-(i-3)} + \ldots + 2^{n_{i-1}-n_i}.$$

Аддитивную сложность вычисления разностей $h_{i,l}$ теперь можно оценить как

$$\sum_{i=2}^{s} 2^{n_i}\sigma_i = \sum_{i=2}^{s}(2^{n_1-(i-2)} + 2^{n_2-(i-3)} + \ldots + 2^{n_{i-1}}) <$$
$$< (2^{n_1} + 2^{n_1-1} + \ldots) + (2^{n_2} + 2^{n_2-1} + \ldots) + \ldots + (2^{n_s} + 2^{n_s-1} + \ldots) < 2n.$$

Число делений на 2 оценивается как

$$\sum_{i=2}^{s}(i-1)2^{n_i} = 2^{n_2} + 2 \cdot 2^{n_3} + 3 \cdot 2^{n_4} + \ldots < 2(n - 2^{n_1}).$$

Заключительная часть схемы такая же, как в п. *а*. ■

2. Троичный метод умножения

2.1. Основные результаты

Здесь, если явно не оговаривается иное, будем полагать $n = 2\,(3^{n_1} + 3^{n_2} + \ldots + 3^{n_s})$, где $n_1 > n_2 > \ldots > n_s$. Заметим, что $n \in [2 \cdot 3^{n_1},\, 3^{n_1+1})$.

Рассмотрим способ использования схем для умножения многочленов по модулям $x^{2\cdot 3^k}+x^{3^k}+1$ с коэффициентами над кольцом, в котором обратим элемент 3. Сложность, глубину и объем памяти таких схем будем обозначать через $M(k)=$ $=M_A(k)+M_3(k)+M_C(k)+M_N(k)$, $d_M(k)$ и $v_M(k)$, где $M_A(k)$ обозначает число аддитивных операций; $M_3(k)$ — число умножений на степени тройки; $M_C(k)$ — число прочих скалярных умножений; $M_N(k)$ — число нескалярных умножений.

Из лемм 5 и 6, приведенных ниже, вытекает

Теорема 3. Для умножения многочленов суммарной степени не выше $n-1$ можно построить схему

а) из $5{,}5n-5\cdot 3^{n_1}+\sum\limits_i M_A(n_i)$ аддитивных операций, $1{,}5n-3^{n_1+1}+\sum\limits_i M_3(n_i)$, $\sum\limits_i M_C(n_i)$ и $\sum\limits_i M_N(n_i)$ умножений на степени тройки, прочие константы и нескалярных умножений соответственно, и глубины $4n_1+\max\limits_i\{d_M(n_i)-2n_i\}-2n_s+s+1$. В случае кольца характеристики 2 справедлива на $0{,}5n$ меньшая оценка аддитивной сложности;

б) из $15{,}5n-19\cdot 3^{n_1}+\sum\limits_i M_A(n_i)$ аддитивных операций, $4{,}75n-8{,}5\cdot 3^{n_1}+\sum\limits_i M_3(n_i)$, $\sum\limits_i M_C(n_i)$ и $\sum\limits_i M_N(n_i)$ умножений на степени тройки, прочие константы и нескалярных умножений соответственно, и объема памяти $2n+\max\limits_i\{v_M(n_i)-4\cdot 3^{n_i}\}$.

Следствие 3. Пусть $M(k)=f(2\cdot 3^k)$, где для любых $x,y\geqslant 1$ справедливо $f(x+y)\geqslant f(x)+f(y)$. Пусть также $d_M(k)-2k\leqslant d_M(l)-2l$ и $v_M(k)-4\cdot 3^k\leqslant v_M(l)-4\cdot 3^l$ при любых $k\leqslant l$. Тогда для умножения многочленов суммарной степени не выше $n-1$ можно построить схему

а) сложности $f(n)+7n-8\cdot 3^{n_1}$ и глубины $d_M(n_1)+3n_1+2$, а в случае кольца характеристики 2 для сложности схемы справедлива оценка $f(n)+5(n-3^{n_1})$;

б) сложности $f(n)+20{,}25n-27{,}5\cdot 3^{n_1}$ и объема памяти $2n+v_M(n_1-4\cdot 3^{n_1})$, а в случае кольца характеристики 2 сложность схемы не превосходит $f(n)+15{,}5n-19\cdot 3^{n_1}$.

В общем случае, а именно для $n\in\bigcup\limits_i[3^i,\,2\cdot 3^i)$, оценку сложности вида $(1+\mathrm{o}(1))f(n)$, не модифицируя базовый алгоритм умножения, получить пока не удается. Но можно доказать оценку $(4/3+\mathrm{o}(1))f(n)$: справедливо

Следствие 4. Пусть в условиях следствия 3 дополнительно выполняется $f(x)/x\to\infty$ при $x\to\infty$ и $f(x)=x^{\mathrm{O}(1)}$. Тогда для умножения многочленов суммарной степени не выше $n-1$ можно построить схему сложности не более

$$\begin{cases}(1+\mathrm{o}(1))f(n), & 2\cdot 3^k\leqslant n<3^{k+1},\\ \bigl(2-3^k/n+\mathrm{o}(1)\bigr)\,f(n), & 3^k\leqslant n<3^{k+1}/2,\\ \bigl(2\cdot 3^k/n+\mathrm{o}(1)\bigr)\,f(n), & 3^{k+1}/2<n<2\cdot 3^k\end{cases}$$

и глубины не более $d_M(\lfloor\log_3(2n+\mathrm{o}(n))\rfloor-1)+\mathrm{o}(\log n)$.

Доказательство. В первом случае конструкция та же, что и в следствии 2. В третьем случае используется схема умножения многочленов суммарной степени не более $2\cdot 3^k-1$.

В случае $n\in[3^k,\,3^{k+1}/2)$ рассмотрим ближайшее сверху число n', кратное $2\cdot 3^{k-\alpha(n)}$, где $\alpha(n)$ — медленно растущая натуральная функция. Если $n'>3^{k+1}/2$, то действуем, как в третьем случае.

Иначе, если $n'<3^{k+1}/2$, вычислим произведение по модулям $x^{2\cdot 3^i}+x^{3^i}+1$, где $i=k-1,\ldots,k-\alpha(n)$, схемой сложности $f(3^k-3^{k-\alpha(n)})+\mathrm{O}(n)$ и глубины $\mathrm{O}(\alpha(n))$ методом теоремы 3.

Фактически это дает нам $3^k - 3^{k-\alpha(n)}$ линейных уравнений на коэффициенты искомого произведения. Для получения оставшихся $n'' = n' - 3^k + 3^{k-\alpha(n)}$ уравнений заметим, что $n'' = L + U$, где

$$2L,\, 2U \in \{0\} \cup \left(\bigcup_{j=1}^{\alpha(n)} [2 \cdot 3^{k-j},\, 3^{k-j+1}) \right) \cap \{2 \cdot 3^{k-\alpha(n)}\mathbb{N}\}.$$

Действительно, любое натуральное число можно представить в виде суммы двух чисел, запись которых в системе счисления с основанием 3 состоит только из нулей и единиц. Все такие числа содержатся в множестве $\{0\} \cup \bigcup\limits_{j=1}^{\infty} [3^j,\, 3^{j+1}/2)$.

Перемножая отдельно младшие части многочленов суммарной степени $2L - 2$ и старшие части суммарной степени $2U - 2$, определяем L младших и U старших коэффициентов произведения. Это выполняется со сложностью $f(2L) + f(2U) + \mathrm{O}(L + U)$ и глубиной $\mathrm{O}(\alpha(n))$, согласно теореме 3.

Далее при помощи леммы 5 находим остатки от деления известной части искомого произведения на многочлены $x^{2\cdot 3^i} + x^{3^i} + 1$, $i = k-1, \ldots, k-\alpha(n)$, откуда находим остатки от деления неизвестной части $a(x)x^L$, где $\deg a < n' - L - U$, на те же многочлены. Все это выполняется со сложностью $\mathrm{O}(n)$ и глубиной $\mathrm{O}(\alpha(n))$.

Многочлен $f(x)$ восстанавливается слегка модифицированным методом леммы 6 (см. ниже; эту модификацию несложно построить) также со сложностью $\mathrm{O}(n)$ и глубиной $\mathrm{O}(\alpha(n))$. Следствие доказано. ■

Конструкции и оценки сложностных характеристик схем умножения по модулю $x^{2\cdot 3^k} + x^{3^k} + 1$ приводятся, например, в [5, 6, 14–16].

2.2. Вспомогательные утверждения

Пусть $a(x) = \sum\limits_{l=0}^{n-1} a_l x^l$. Положим формально $a_l = 0$ при $l \geqslant n$. Введем обозначения

$$a(x) \bmod (x^{3^k} - 1) = \sum_{l=0}^{3^k-1} a_{k,l} x^l, \quad a(x) \bmod (x^{2\cdot 3^k} + x^{3^k} + 1) = \sum_{r=0}^{1} \sum_{l=0}^{3^k-1} b_{k,r,l} x^l.$$

Коэффициенты $a_{k,l}$ и $b_{k,r,l}$ связывают простые соотношения

$$a_{k,l} = a_{k+1,l} + a_{k+1,3^k+l} + a_{k+1,2\cdot 3^k+l}, \quad b_{k,r,l} = a_{k+1,r3^k+l} - a_{k+1,2\cdot 3^k+l}.$$

На них основана

Лемма 4. Пусть $n_{i+1} < k \leqslant n_i$. Справедливы следующие формулы:

$$a_{k,l} = \sum_{j_i=0}^{3^{n_i-k}-1} \Big(b_{n_i,\,0,\,j_i 3^k+l} + b_{n_i,\,1,\,j_i 3^k+l} +$$
$$+3 \sum_{j_{i-1}=0}^{3^{n_{i-1}-n_i-1}-1} \Big(b_{n_{i-1},\,0,\,(3j_{i-1}+2)3^{n_i}+j_i 3^k+l} + b_{n_{i-1},\,1,\,(3j_{i-1}+2)3^{n_i}+j_i 3^k+l} +$$
$$+3 \sum_{j_{i-2}=0}^{3^{n_{i-2}-n_{i-1}-1}-1} \Big(b_{n_{i-2},\,0,\,(3j_{i-2}+2)3^{n_{i-1}}+(3j_{i-1}+2)3^{n_i}+j_i 3^k+l} +$$
$$+ b_{n_{i-2},\,1,\,(3j_{i-2}+2)3^{n_{i-1}}+(3j_{i-1}+2)3^{n_i}+j_i 3^k+l} + \ldots$$

$$\ldots+3\sum_{j_1=0}^{3^{n_1-n_2-1}-1}\Big(b_{n_1,\,0,\,(3j_1+2)3^{n_2}+\ldots+(3j_{i-1}+2)3^{n_i}+j_i3^k+l}+$$

$$+\,b_{n_1,\,1,\,(3j_1+2)3^{n_2}+\ldots+(3j_{i-1}+2)3^{n_i}+j_i3^k+l}\Big)\ldots\Big)\Big)\Big)+3^i a_{2(3^{n_1}+\ldots+3^{n_i})+l};$$

$$b_{n_i,r,l}=\sum_{j=0}^{1}(-1)^j\Bigg(\sum_{j_{i-1}=0}^{3^{n_{i-1}-n_i-1}-1}\Big(b_{n_{i-1},\,0,\,(3j_{i-1}+r+j(2-r))3^{n_i}+l}+$$

$$+b_{n_{i-1},\,1,\,(3j_{i-1}+r+j(2-r))3^{n_i}+l}+$$

$$+3\sum_{j_{i-2}=0}^{3^{n_{i-2}-n_{i-1}-1}-1}\Big(b_{n_{i-2},\,0,\,(3j_{i-2}+2)3^{n_{i-1}}+(3j_{i-1}+r+j(2-r))3^{n_i}+l}+$$

$$+b_{n_{i-2},\,1,\,(3j_{i-2}+2)3^{n_{i-1}}+(3j_{i-1}+r+j(2-r))3^{n_i}+l}+\ldots$$

$$\ldots+3\sum_{j_1=0}^{3^{n_1-n_2-1}-1}\Big(b_{n_1,\,0,\,(3j_1+2)3^{n_2}+\ldots+(3j_{i-2}+2)3^{n_{i-1}}+(3j_{i-1}+r+j(2-r))3^{n_i}+l}+$$

$$+b_{n_1,\,1,\,(3j_1+2)3^{n_2}+\ldots+(3j_{i-2}+2)3^{n_{i-1}}+(3j_{i-1}+r+j(2-r))3^{n_i}+l}\Big)\ldots\Big)\Big)\Big)+$$

$$+3^{i-1}\left(a_{2(3^{n_1}+\ldots+3^{n_i-1})+r3^{n_i}+l}-a_{2(3^{n_1}+\ldots+3^{n_i})+l}\right).$$

Доказательство. Доказательство полностью аналогично доказательству леммы 1, только используем соотношения

$$a_{k,l}=\begin{cases}a_{k+1,l}+a_{k+1,3^k+l}+a_{k+1,\,2\cdot3^k+l}, & k\notin\{n_i:i=1,\ldots,s\},\\ b_{k,0,l}+b_{k,1,l}+3a_{k+1,\,2\cdot3^k+l}, & k\in\{n_i:i=1,\ldots,s\},\end{cases}$$

учитывая, что $a_{n_1+1,l}=a_l$ и, кроме того, $a_l=0$ при $l\geqslant n$. Вторая формула получается из первой как $b_{n_i,r,l}=a_{n_i+1,\,r3^{n_i}+l}-a_{n_i+1,\,2\cdot3^{n_i}+l}$. ■

Лемма 5. Пусть $m\leqslant 2n$, а n и n_i — такие, как в теореме 3. Тогда приведение многочлена степени не выше $m-1$ по модулям $x^{2\cdot3^{n_i}}+x^{3^{n_i}}+1$, $i=1,\ldots,s$, может быть выполнено схемой

а) из $2(m-1)$ аддитивных элементов и глубины $2(n_1-n_s)+1$, причем коэффициенты остатка от деления на $x^{2\cdot3^{n_i}}+x^{3^{n_i}}+1$ вычисляются на глубине $2(n_1-n_i+1)$. Кроме того, в случае кольца характеристики 2 для сложности схемы справедлива оценка $1{,}5(m-1)$;

б) из $6n-8\cdot3^{n_1}$ аддитивных элементов, $13n/8-11\cdot3^{n_1}/4$ умножений на степени тройки и объема памяти n.

Доказательство. Все суммы $a_{k,l}$, где $k=n_s+1,\ldots,n_1$, можно вычислить одной схемой сложности не выше $m-3^{n_s+1}$, в которой глубина вычисления $a_{k,l}$ не превосходит $2(n_1+1-k)$.

Коэффициенты $b_{n_i,r,l}$ остатка от деления на $x^{2\cdot3^{n_i}}+x^{3^{n_i}}+1$ получаются присоединением к построенной схеме элементов вычитания, расположенных на глубине 1 относительно $a_{n_i+1,l}$, в количестве

$$\begin{cases}0, & m\leqslant 2\cdot3^{n_i},\\ 2(m-2\cdot3^{n_i}), & 2\cdot3^{n_i}\leqslant m\leqslant 3^{n_i+1},\\ 2\cdot3^{n_i}, & m\geqslant 3^{n_i+1}.\end{cases}$$

Число вычитаний в двух последних случаях можно также оценить сверху как $m - 3^{n_i}$. Поскольку $3^{n_i} > 2(3^{n_{i+1}} + \ldots + 3^{n_s})$, для числа вычитаний (при подходящем i) получаем оценку

$$m - 3^{n_i} + 2\left(3^{n_{i+1}} + \ldots + 3^{n_s}\right) \leqslant m - 1,$$

откуда следует оценка $2(m-1)$ для сложности всей схемы.

Если характеристика равна 2, то один из двух коэффициентов в каждой паре $b_{n_i,0,l}$, $b_{n_i,1,l}$ можно использовать при вычислении $a_{n_i,l}$ (кроме случая $i = s$). Для сложности схемы в этом случае имеем оценку $m - 3^{n_s+1} + 0{,}5(m-1) + 3^{n_s} < 1{,}5(m-1)$. П. *а* доказан.

Доказательство п. *б* достаточно провести для случая $m = n$. Перестроим схему из предыдущего пункта так, чтобы в ней явно вычислялись только те суммы $a_{k,l}$, для которых «достаточно памяти».

Обозначим $L_i = 2(3^{n_i} + \ldots + 3^{n_s})$. При любом $k = n_1, \ldots, n_s + 1$ мы явно вычисляем суммы $a_{k,0}, \ldots, a_{k,L_i-1}$, где $n_{i-1} \geqslant k > n_i$, учитывая, что $n - L_i$ прочих «ячеек памяти» отводится под хранение коэффициентов $b_{n_j,r,l}$, где $j < i$.

Все вычисление удобно разбить на этапы, нумеруя их числами от n_1 до n_s в порядке убывания. На этапе k вычисляются суммы $a_{k,0}, \ldots, a_{k,L_i-1}$, и если $k \in \{n_i : i = 1, \ldots, s\}$, то вычисляются также коэффициенты $b_{k,r,l}$, $r \in \{0, 1\}$, $l = 0, \ldots, 3^k - 1$.

Если сумма $a_{k,l}$ не вычисляется явно, то вместо нее используется правая часть формулы (обозначим ее $\psi_{k,l}$) из леммы 4, выражающая ее через известные коэффициенты $b_{n_j,q,t}$, где $n_j \geqslant k$. Заметим, что поскольку $l \geqslant L_{i+1}$ при $n_i \geqslant k > n_{i+1}$, последнее слагаемое $a_{2(3^{n_1}+\ldots+3^{n_i})+l}$ в $\psi_{k,l}$ равно нулю.

Обозначим через ρ_k число переменных в формуле $\psi_{k,l}$ (это число не зависит от l, что следует из вида формулы). Тогда прибавление (вычитание) $\psi_{k,l}$ выполняется за ρ_k аддитивных операций и i умножений на степени тройки без дополнительной памяти, где $n_i \geqslant k > n_{i+1}$.

Оценим сложность схемы. Рассмотрим этап с номером k, где $n_{i-1} > k > n_i$. Доступны коэффициенты $a_{k+1,0}, \ldots, a_{k+1,L_i-1}$, остальные $a_{k+1,l}$ выражаются формулами $\psi_{k+1,l}$. Заметим, что $L_i < 3^{n_i+1} \leqslant 3^k$. Тогда каждый из коэффициентов $a_{k,l}$, где $l < L_i$, вычисляется как $a_{k+1,l} + \psi_{k+1,3^k+l} + \psi_{k+1,2\cdot3^k+l}$ со сложностью $2\rho_{k+1}$ аддитивных операций и $i - 1$ умножений на степени тройки (умножения на степени тройки при прибавлении двух формул типа ψ можно совместить). Сложность этапа оценивается как $2L_i\rho_{k+1}$ аддитивных операций и $(i-1)L_i$ умножений на степени тройки.

Рассмотрим случай $k = n_i$. Доступны коэффициенты $a_{k+1,0}, \ldots, a_{k+1,L_i-1}$. При этом $L_i = 2 \cdot 3^k + L_{i+1}$. Сначала вычислим все $b_{k,r,l}$ по формулам $a_{k+1,r3^k+l} - a_{k+1,2\cdot3^k+l}$ при $l < L_{i+1}$ и $a_{k+1,r3^k+l} - \psi_{k+1,2\cdot3^k+l}$ при прочих l. При этом коэффициентами $b_{k,r,l}$ перезаписываются $a_{k+1,l}$, $l < 2 \cdot 3^k$. Затем вычислим $a_{k,l}$ для $l < L_{i+1}$ по формулам $3a_{k+1,2\cdot3^k+l} + b_{k,0,l} + b_{k,1,l}$. Сложность этапа оценивается как $4L_{i+1} + 2(3^k - L_{i+1})\rho_{k+1}$ аддитивных операций и $2(i-1)3^k - (2i-3)L_{i+1}$ умножений на степени тройки. Эта оценка справедлива и в случае $k = n_1$, так как можно положить $\rho_{n_1+1} = 0$.

Найдем ρ_k. Из вида формулы леммы 4 непосредственно следует, что

$$\rho_{n_i} = 2\left(3^{n_1-n_i-(i-1)} + 3^{n_2-n_i-(i-2)} + \ldots + 3^{n_{i-1}-n_i-1} + 1\right), \tag{3}$$

а $\rho_k = 3^{n_i-k}\rho_{n_i}$ при $n_{i+1} < k < n_i$.

Оценим суммарную аддитивную сложность вычислений. Сумма сложностей этапов $n_i - 1, \ldots, n_{i+1}$ не превосходит

$$C_i = 2L_{i+1}\rho_{n_i}(1 + 3 + \ldots + 3^{n_i - n_{i+1} - 2}) + 4L_{i+2} + 2(3^{n_{i+1}} - L_{i+2})3^{n_i - n_{i+1} - 1}\rho_{n_i} \leqslant$$
$$\leqslant 2\rho_{n_i}3^{n_i - n_{i+1} - 1}\left(\frac{1}{2}L_{i+1} - L_{i+2} + 3^{n_{i+1}}\right) - \rho_{n_i}L_{i+1} + 4L_{i+2} =$$
$$= 2\rho_{n_i}3^{n_i - n_{i+1} - 1}\left(2 \cdot 3^{n_{i+1}} - \frac{1}{2}L_{i+2}\right) - \rho_{n_i}L_{i+1} + 4L_{i+2} =$$
$$= \rho_{n_i}(4 \cdot 3^{n_i - 1} - L_{i+2}3^{n_i - n_{i+1} - 1} - L_{i+1}) + 4L_{i+2}.$$

При $i \geqslant 1$ последнее выражение не превосходит $4 \cdot 3^{n_i - 1}\rho_{n_i}$, если учесть, что $L_{i+1} > 3L_{i+2}$ и $\rho_{n_i} \geqslant 1$.

Для сложности всех этапов, кроме этапа n_1, используя (3), получаем оценку

$$C_1 + \ldots + C_{s-1} = 4\sum_{i=1}^{s-1} 3^{n_i - 1}\rho_{n_i} = 8\sum_{i=1}^{s-1}(3^{n_1 - i} + 3^{n_2 - (i-1)} + \ldots + 3^{n_i - 1}) <$$
$$< 8\left((3^{n_1 - 1} + 3^{n_1 - 2} + \ldots) + (3^{n_2 - 1} + 3^{n_2 - 2} + \ldots) + \ldots + (3^{n_s - 1} + 3^{n_s - 2} + \ldots)\right) < 2n.$$

Окончательно, оценивая сложность этапа n_1 как $4L_2 = 4(n - 2 \cdot 3^{n_1})$, получаем утверждение п. *б* в части аддитивной сложности.

Число умножений на степени тройки на этапах $n_i, \ldots, n_{i+1} + 1$ оценивается как

$$D_i = (n_i - n_{i+1} - 1)iL_{i+1} + 2(i-1)3^{n_i} - (2i-3)L_{i+1} =$$
$$= i(n_i - n_{i+1})L_{i+1} + (i-1)(L_i - 4L_{i+1}).$$

Сумму первых слагаемых можно оценить как

$$\sum_{i=1}^{s} i(n_i - n_{i+1})L_{i+1} \leqslant \sum_{i=1}^{s} i3^{n_i - n_{i+1} - 1}(3^{n_{i+1}+1}/2) = \frac{1}{2}\sum_{i=1}^{s} i3^{n_i} <$$
$$< \frac{1}{2}\left((3^{n_1} + 3^{n_2} + \ldots) + (3^{n_2} + 3^{n_3} + \ldots) + \ldots\right) <$$
$$< \frac{1}{2}\left(\frac{n}{2} + (n/2 - 3^{n_1}) + \frac{1}{3}(n/2 - 3^{n_1}) + \ldots\right) = \frac{n}{4} + \frac{3}{4}(n/2 - 3^{n_1}).$$

Сумму вторых слагаемых оценим грубо как

$$(L_2 - 4L_3) + 2(L_3 - 4L_4) + 3(L_4 - 4L_5) + \ldots \leqslant L_2 = n - 2 \cdot 3^{n_1}.$$

Складывая последние две оценки, завершаем доказательство п. *б*. ■

Лемма 6. Восстановление многочлена степени $n - 1$ по заданным остаткам от деления на многочлены $x^{2 \cdot 3^{n_i}} + x^{3^{n_i}} + 1$, $i = 1, \ldots, s$, может быть выполнено схемой

а) из $3{,}5n - 5 \cdot 3^{n_1}$ аддитивных элементов, $1{,}5(n - 2 \cdot 3^{n_1})$ умножений на степени тройки и глубины $2(n_1 - n_s) + s + 1$;

б) из $3{,}5n - 3^{n_1+1}$ аддитивных элементов, $1{,}5(n - 2 \cdot 3^{n_1})$ делений на 3 и объема памяти n.

Доказательство. Из леммы 4 выразим разности

$$h_{i,r,l} = a_{2(3^{n_1} + \ldots + 3^{n_i - 1}) + r3^{n_i} + l} - a_{2(3^{n_1} + \ldots + 3^{n_i}) + l}, \tag{4}$$

где $i = 1, \ldots, s-1$, $r \in \{0, 1\}$ и $l = 0, \ldots, L_i - 1$, через коэффициенты $b_{n_j,q,t}$. Заметим, что вычитаемый коэффициент равен нулю при $l \geqslant L_{i+1}$.

Для построения схемы из п. *а* вычисляем вспомогательные величины $c_{i,l}$ и $d_{i,l}$, определяемые равенствами

$$d_{1,l} = b_{n_1,0,l} + b_{n_1,1,l}, \quad c_{i,l} = \sum_{j=0}^{3^{n_{i-1}-n_i-1}-1} d_{i-1,\,j3^{n_i+1}+l}, \quad d_{i,l} = b_{n_i,0,l} + b_{n_i,1,l} + 3c_{i,2\cdot 3^{n_i}+l},$$

где $i \geqslant 2$. Искомые коэффициенты $h_{i,r,l}$ при $i \geqslant 2$ выражаются формулами $h_{i,r,l} = 3^{1-i}(b_{n_i,r,l} - c_{i,r3^{n_i}+l} + c_{i,2\cdot 3^{n_i}+l})$, а при $i = 1$ просто $h_{1,r,l} = b_{n_1,r,l}$.

Сложность вычислений, заключенная в формулах для $c_{i,l}$, $i = 2, \ldots, s$ и $l = 0, \ldots, 3^{n_i+1} - 1$, оценивается как

$$\sum_{i=2}^{s} 3^{n_i+1}(3^{n_{i-1}-n_i-1} - 1) = \sum_{i=2}^{s}(3^{n_{i-1}} - 3^{n_i+1}) < n/2 - 3(n/2 - 3^{n_1}) = 3^{n_1+1} - n.$$

Сложность, заключенная в формулах для $d_{i,l}$, $i = 1, \ldots, s-1$ и $l = 0, \ldots, 3^{n_i} - 1$, оценивается как $3^{n_1} + 2(3^{n_2} + \ldots + 3^{n_{s-1}}) < n - 3^{n_1}$ аддитивных операций и $3^{n_2} + \ldots + 3^{n_{s-1}} < n/2 - 3^{n_1}$ умножений на 3. Для завершения вычисления $h_{i,r,l}$ нужно выполнить еще $4(3^{n_2} + \ldots + 3^{n_s}) = 2(n - 2 \cdot 3^{n_1})$ аддитивных операций и $n - 2 \cdot 3^{n_1}$ умножений на степени тройки.

Без труда проверяется, что $c_{i,l}$ при этом вычисляется на глубине $2(n_1 - n_i) - 1$ и, следовательно, $h_{i,r,l}$ — на глубине $2(n_1 - n_i) + 2$.

По набору $h_{i,r,l}$ несложно восстанавливаются коэффициенты многочлена $a(x)$. Коэффициенты $a_{2(3^{n_1}+\ldots+3^{n_{i-1}})+r3^{n_i}+l}$ совпадают с $h_{i,r,l}$ при $l \geqslant L_{i+1}$. В частности, в случае $i = s$ известны все коэффициенты. Это позволяет последовательно в порядке убывания i определить недостающие коэффициенты $a_{2(3^{n_1}+\ldots+3^{n_{i-1}})+r3^{n_i}+l}$, $l = 0, \ldots, L_{i+1} - 1$, из формул (4) со сложностью

$$L_s + L_{s-1} + \ldots + L_2 = 2(3^{n_2} + 2 \cdot 3^{n_3} + 3 \cdot 3^{n_4} + \ldots) < 1{,}5(n - 2 \cdot 3^{n_1})$$

аддитивных операций. Глубина этих вычислений не превосходит $s - 1$.

Складывая все оценки, получаем утверждение п. *а*.

Для доказательства п. *б* построим схему, последовательно в порядке убывания i преобразующую каждый коэффициент $b_{n_i,r,l}$ по формуле леммы 4 в соответствующую разность $h_{i,r,l}$.

Несложно видеть, что преобразование каждого коэффициента $b_{n_i,r,l}$ выполняется за τ_i аддитивных операций и $i - 1$ делений на 3, где τ_i — число коэффициентов $b_{n_j,q,t}$ в правой части второй формулы леммы 4. Непосредственно проверяется, что

$$\tau_i = 4\left(3^{n_1-n_i-(i-1)} + 3^{n_2-n_i-(i-2)} + \ldots + 3^{n_{i-1}-n_i-1}\right).$$

Аддитивную сложность вычисления разностей $h_{i,r,l}$ теперь можно оценить как

$$\sum_{i=2}^{s} 2 \cdot 3^{n_i}\tau_i = \sum_{i=2}^{s} 8\left(3^{n_1-(i-1)} + 3^{n_2-(i-2)} + \ldots + 3^{n_{i-1}-1}\right) <$$
$$< 8\left((3^{n_1-1} + 3^{n_1-2} + \ldots) + (3^{n_2-1} + 3^{n_2-2} + \ldots) + \ldots + (3^{n_s-1} + 3^{n_s-2} + \ldots)\right) < 2n.$$

Число делений на 3 оценивается как

$$\sum_{i=2}^{s} 2(i-1)3^{n_i} = 2\left(3^{n_2} + 2 \cdot 3^{n_3} + 3 \cdot 3^{n_4} + \ldots\right) < 1{,}5(n - 2 \cdot 3^{n_1}).$$

Заключительная часть схемы такая же, как в п. *а*. ■

ЛИТЕРАТУРА

1. *Van der Hoeven J.* The truncated Fourier transform and applications // Proc. ISSAC 2004 (Santander, Spain). NY: ACM Press, 2004. P. 290–296.
2. *Harvey D. and Roche D. S.* An in-place truncated Fourier transform and application to polynomial multiplication // Proc. ISSAC 2010 (Munich, Germany). NY: ACM Press, 2010. P. 325–329.
3. *Schönhage A.* Schnelle multiplikation von polynomen über körpern der charakteristik 2 // Acta Inf. 1977. V. 7. P. 395–398.
4. *Cantor D. and Kaltofen E.* On fast multiplication of polynomials over arbitrary algebras // Acta Inf. 1991. V. 28. No. 7. P. 693–701.
5. *Bernstein D. J.* Fast multiplication and its applications // Algorithmic Number Theory, MSRI Publ. 2008. V. 44. P. 325–384.
6. *Von zur Gathen J. and Gerhard J.* Modern computer algebra. Cambridge: Cambridge University Press, 1999. 768 p.
7. *Лупанов О. Б.* Асимптотические оценки сложности управляющих систем. М.: Изд-во Моск. ун-та, 1984. 138 с.
8. *Яблонский С. В.* Введение в дискретную математику. М.: Наука, 1986. 384 с.
9. *Cooley J. and Tukew J.* An algorithm for the machine calculation of complex Fourier series // Math. Comp. 1965. V. 19. P. 297–301.
10. *Schönhage A.* Asymptotically fast algorithms for the numerical multiplication and division of polynomials with complex coefficients // Proc. EuroCAM-82 (Marseille, France). LNCS. V. 144. Berlin; Heidelberg; NY: Springer, 1982. P. 3–15.
11. *Сергеев И. С.* Регуляризация некоторых оценок сложности умножения многочленов // Материалы VII молодежной научной школы по дискретной математике и ее приложениям (Москва, 2009 г.). Ч. II. М.: Изд-во Института прикладной математики РАН, 2009. С. 26–32.
12. *Van der Hoeven J.* Notes on the truncated Fourier transform // Tech. Report. Univ. Paris-Sud, Orsay, France, 2005.
13. *Crandall R. and Fagin B.* Discrete weighted transforms and large-integer arithmetic // Math. Comput. 1994. V. 62. P. 305–324.
14. *Mateer T.* Fast Fourier algorithms with applications // Ph. D. Thesis. Clemson University, 2008.
15. *Гашков С. Б., Сергеев И. С.* Алгоритмы быстрого преобразования Фурье // Дискретная математика и ее приложения. Ч. V. М.: Изд-во Института прикладной математики РАН, 2009. С. 3–23.
16. *Гашков С. Б., Сергеев И. С.* О сложности и глубине булевых схем для умножения и инвертирования в некоторых полях $GF(2^n)$ // Вестник МГУ. Сер. 1. Математика. Механика. 2009. № 4. С. 3–7.