МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

имени М.В.ЛОМОНОСОВА

МЕХАНИКО-МАТЕМАТИЧЕСКИЙ ФАКУЛЬТЕТ

Отделение МАТЕМАТИКИ

Кафедра ТЕОРИИ ЧИСЕЛ

Быстрое умножение целых чисел.

Fast multiplication of integer numbers.

Дипломная работа

студентки 512 группы

Трегубовой Анны Андреевны

Научный руководитель:

Член-корр. РАН, профессор, доктор физико-математических наук

Нестеренко Юрий Валентинович

Москва 2015 год

Решения многих задач теории чисел активно применяются и используются в различных областях математики, информатики, криптографии и других науках. Одной из таких известных задач является задача умножения целых чисел. Школьники ее решают и легко находят ответ применяя метод "умножения в столбик". Суть метода в том, чтобы каждую цифру одного числа умножать на каждую цифру другого числа, то есть с каждой цифрой провести k операций, следовательно, получается сложность $O(k^2)$, где k - длина чисел. Этому методу уже порядка 4000 лет, его достоинства в простоте и наглядности, и, казалось, трудно придумать метод эффективнее. В результате в 1956 году А. Н. Колмогоров сформулировал гипотезу, что нижняя оценка для сложности умножения целых чисел при любом методе умножения есть также величина порядка k^2.

Однако уже в 1960 году начали придумывать "быстрые"алгоритмы. А. Карацуба предложил новый метод умножения двух k-значных чисел с оценкой сложности $O(k^{\log_2 3})$.

Идея заключается в разбиении чисел a и b длины $2k$ на сумму блоков длины k. Т.е. $a = a_1 \cdot T + a_2$ и $b = b_1 \cdot T + b_2$, где a_1, a_2, b_1, b_2 - длины k, T -величина сдвига, если рассматривать числа в двоичной записи, то можно взять $T = 2^k$. Тогда

$$ab = (a_1 \cdot 2^k + a_2)(b_1 \cdot 2^k + b_2) = a_1 b_1 \cdot 2^{2k} + (a_1 b_2 + a_2 b_1)2^k + a_2 b_2$$

Получаем, что нужно сделать 4 умножения $a_1 b_1, a_2 b_1, a_1 b_2, a_2 b_2$. Карацуба предложил посмотреть на это выражение по-другому и переписать его в виде:

$$ab = a_1 b_1 \cdot 2^{2k} + ((a_1 + a_2)(b_1 + b_2) - a_1 b_1 - a_2 b_2) \cdot 2^k + a_2 b_2.$$

Здесь нужно сделать всего 3 умножения: $a_1 b_1, (a_1 + a_2)(b_1 + b_2), a_2 b_2$, при этом, длина каждого из этих чисел не более $k + 1$. Применяя полученный алгоритм рекурсивно, получаем, что сложность вычисления снижается до $k^{log_2 3}$.

Впоследствии метод Карацубы был обобщён до парадигмы «Разделяй и властвуй», то есть рекурсивного разбиения решаемой задачи на две или более подзадачи того же типа, но меньшего размера, и комбинирования их решений для получения ответа к исходной задаче; разбиения выполняются до тех пор, пока все подзадачи не окажутся элементарными. Затем последовал ряд работ, основанных на идее Карацубы. В них авторы пытаются улучшить оценку за счет разбиения на другие блоки. Возникает вопрос, какова оптимальная оценка сложности, то есть минимальная степень при k?

Можно заметить, что оценкой снизу является $O(k)$, так как нужно обработать каждый разряд числа, а разрядов k. Но насколько близко можно подойти к этой оценке?

Многие пытались решить эту задачу и, наконец, Тоом в своей работе показал, что можно предъявить алгоритм умножения целых чисел с оценкой сложности $O(k^{1+\varepsilon}), \forall \varepsilon > 0$. Его алгоритм заключается в разбиении чисел a и b на h блоков, каждый из которых длины l, таким образом, для чисел получаем выражения:

$$a = \sum_{i=0}^{h-1} a_i 2^{il}, b = \sum_{i=0}^{h-1} b_i 2^{il},\ 0 \leqslant a_i, b_i < 2^l, k = lh. \tag{1}$$

То есть a и b можно представить в виде многочленов

$$f(x) = \sum_{i=0}^{h-1} a_i \cdot x^i, g(x) = \sum_{i=0}^{h-1} b_i \cdot x^i, \tag{2}$$

и очевидно, что $a = f(2^l), b = g(2^l)$. Тогда

$$ab = f(2^l) \cdot g(2^l) = \sum_{k=0, k=i+j}^{2h-1} a_i b_j \cdot 2^{(i+j)l} = \sum_{k=0}^{2h-1} c_k \cdot 2^{kl} = c(2^l). \tag{3}$$

Нужно вычислить коэффициенты c_k. Для этого отдельно вычисляем значения многочленов $f(x)$ и $g(x)$ в точках $-h+1, -h+2, ...h-1, h-2$. Далее восстанавливаем коэффициенты c_k и вычисляем $c(2^l)$, что является ответом задачи умножения.

В качестве примера, показывающего как получается ускорение, рассмотрим случай $h = 3$. Тогда нужно вычислить значения $f(x)$ и $g(x)$ в точках $0, -1, 1, -2, 2$. В среднем столбце таблицы показано, чему равно выражение (2) через коэффициенты $f(x)$ и $g(x)$, в последнем столбце - выражения (3) для коэффициентов c_k.

$x = 0$	$a_0 b_0$	$c(0) = c_0$
$x = 1$	$(a_2 + a_1 + a_0) \cdot (a_2 + a_1 + a_0)$	$c(1) = c_4 + c_3 + c_2 + c_1 + c_0$
$x = -1$	$(a_2 - a_1 + a_0) \cdot (b_2 - b_1 + b_0)$	$c(-1) = c_4 - c_3 + c_2 - c_1 + c_0$
$x = 2$	$(4a_2 + 2a_1 + a_0) \cdot (4b_2 + 2b_1 + b_0)$	$c(2) = 16c_4 + 8c_3 + 4c_2 + 2c_1 + c_0$
$x = -2$	$(4a_2 - 2a_1 + a_0) \cdot (4b_2 - 2b_1 + b_0)$	$c(-2) = 16c_4 - 8c_3 + 4c_2 - 2c_1 + c_0$

Получив линейную систему относительно 5 неизвестных и решая ее, находим все c_k и вычисляем значение многочлена $c(2^l)$.

При выборе подходящих h и l можно достичь нужной оценки $O(k^{1+\varepsilon}), \forall \varepsilon > 0$.

Но на этом результате Тоома развитие задачи не закончилось, и начали появляться новые алгоритмы, улучшающие сложность. Однако, они основаны уже не просто

на разбиении чисел, а используют преобразование Фурье, которое вычисляется методом быстрого преобразования Фурье (БПФ). Сложность БПФ оценивается уже как $O(k \cdot log\, k)$. Об этом подробнее далее.

Сначала введем все необходимые определения.

Определение 1. Пусть R - произвольное ассоциативное коммутативное кольцо с единицей, на котором заданы две операции: сложение и умножение.

Определение 2. Пусть d - некоторое натуральное число. Пусть $\bar{d}$ - элемент кольца, равный сумме d единиц, и $\bar{d}$ имеет обратный, который обозначим $\bar{d}^{-1}$.
Будем говорить, что элемент $g \in \mathrm{R}$ является *примитивным корнем* d-ой степени из 1, если g обладает следующими свойствами:

1. $g \neq 1$;
2. $g^d = 1$, $d = min\{d' \in \mathbb{N} | g^{d'} = 1\}$;
3. $\sum_{i=0}^{d-1} g^{ip} = 0$, для $1 \leq p < d$.

Теперь в данном кольце для $d \in \mathrm{N}$ определим преобразование Фурье.

Определение 3. *Дискретным преобразованием Фурье* называется отображение, сопоставляющее каждому набору $x = (x_0, x_1, ..., x_{d-1}) \in \mathrm{R}^d$ некоторый другой набор $\hat{x} = (\hat{x}_0, \hat{x}_1, ..., \hat{x}_{d-1}) \in \mathrm{R}^d$ такой, что

$$\hat{x}_i = \sum_{j=0}^{d-1} x_j g^{ij}, \qquad i = 0, ..., d-1. \tag{4}$$

По-другому его можно записать в виде: вектор $x = (x_0, x_1, ..., x_{d-1})$ умножается на матрицу $B = (g^{ij}), 0 \leq i, j < d$, в полученном векторе i- ая компонента равняется $\sum_{j=0}^{d-1} x_j g^{ij}$, что соответствует формуле (4). Полученная матрица B имеет вид матрицы Вандермонда:

$$\begin{pmatrix} 1 & 1 & ... & 1 \\ 1 & g & ... & g^{d-1} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & g^{d-1} & ... & g^{(d-1)(d-1)} \end{pmatrix}$$

следовательно, ее определитель равен $\prod_{0 \leqslant j < i \leqslant d-1} (g^i - g^j)$.

Если $g^i = g^j$ для некоторых $0 \leqslant j < i \leqslant d-1$, то $g^{i-j} = 1$, по определению примитивного корня, получаем, что $i = j$- противоречие. Следовательно, определитель не равен нулю и матрица невырождена. Тогда существует матрица B^{-1} - обратная к матрице B. Рассмотрим, как она устроена.

Лемма 1.

Пусть матрица $B = (g^{ij}), 0 \leqslant i, j < d$. Тогда обратная к ней матрица B^{-1} имеет вид $B^{-1} = (\bar{d}^{-1} g^{-ij})$.

Доказательство.

Покажем, что $B \cdot B^{-1} = I_n$.

Для элемента, стоящего на пересечении столбца с номером j и строки с номером i имеем представление:

$$\bar{d}^{-1} \sum_{k=0}^{d-1} g^{ik} g^{-kj}; \tag{5}$$

Возможны 2 случая. Если $i = j$, то выражение (5) имеет вид:

$$\bar{d}^{-1} \sum_{k=0}^{d-1} g^{ik} g^{-kj} = \bar{d}^{-1} \sum_{k=0}^{d-1} g^{(i-j)k} = \bar{d}^{-1} \sum_{k=0}^{d-1} g^0 = 1.$$

Если $i \neq j$, то обозначим $q = i - j$. Тогда

$$\bar{d}^{-1} \sum_{k=0}^{d-1} g^{ik} g^{-kj} = \bar{d}^{-1} \sum_{k=0}^{d-1} g^{qk}, \quad -d < q < d,\ q \neq 0. \tag{6}$$

По определению примитивного корня при $q > 0$ сумма в правой части равенства (6) равна нулю.

При $q < 0$, умножим левую часть на $g^{-q(d-1)}$, меняем индекс суммирования $k' = d - 1 - k$, т.к. g-примитивный, то сумма равна нулю; другими словами:

$$g^{-q(d-1)} \cdot \bar{d}^{-1} \sum_{k=0}^{d-1} g^{qk} = \bar{d}^{-1} \sum_{k=0}^{d-1} g^{(-q)(d-1-k)} = \bar{d}^{-1} \sum_{k'=0}^{d-1} g^{-qk'} = 0, \quad 0 < -q < d.$$

Таким образом, сумма (5) равна 1 только при $i = j$, следовательно, $\bar{d}^{-1} \sum_{k=0}^{d-1} g^{ik} g^{-kj} = \delta_{ij}$. Утверждение леммы доказано.

Следствие 1. *Обратное преобразование Фурье* вектора $\hat{x} = (\hat{x}_0, \hat{x}_1, ..., \hat{x}_{d-1}) \in \mathbb{R}^d$ можно вычислять по формуле:

$$x_i = \bar{d}^{-1} \sum_{j=0}^{d-1} \hat{x}_j g^{-ij}, i = 0, ..., d-1.$$

Для вычисления результата умножения двух целых чисел используется понятие свертки.

Определение 4. Пусть заданы два набора $x = (x_0, x_1, ..., x_{d-1}), y = (y_0, y_1, ..., y_{d-1}) \in \mathrm{R}^d$. *Сверткой* $x * y$ называется набор $z = (z_0, z_1, ..., z_{d-1}) \in \mathrm{R}^d$, такой что

$$z_k = \sum_{i+j \equiv k (mod\, d)} x_i y_j, \qquad k = 0, ..., d-1$$

Если считать строго по формуле, то вычисление свертки для каждого k занимает $O(d)$ операций, значит для всех k получаем $O(d^2)$ операций.
С помощью преобразования Фурье и следующей леммы задачу вычисления свертки можно свести к задаче покомпонентного произведения.

Определение 5. Для наборов $x = (x_0, x_1, ..., x_{d-1}), y = (y_0, y_1, ..., y_{d-1}) \in \mathbb{R}^d$ *покомпонентным произведением* называется набор $x \cdot y = (x_0 y_0, x_1 y_1, ..., x_{d-1} y_{d-1})$.
И имеет место следующая лемма.

Лемма 2.

Пусть $z = x * y$, тогда $\hat{z} = \hat{x} \cdot \hat{y}$.

Доказательство.

Рассмотрим любое $m \in \{0, ..., d-1\}$, тогда для него получим следующее выражение:

$$\sum_{k=0}^{d-1} z_k g^{km} = \sum_{k=0}^{d-1} g^{km} \left(\sum_{i+j \equiv k (mod\, d)} x_i y_j \right)$$

Это равенство выполнено, так как $g^d \equiv 1$ в кольце R, теперь меняем порядок суммирования:

$$\sum_{i=0}^{d-1} x_i g^{mi} \sum_{k=0}^{d-1} g^{m(k-i)} y_{k-i} = \sum_{i=0}^{d-1} x_i g^{mi} \sum_{j=0}^{d-1} y_j g^{mj},$$

Значит, $\hat{z}_m = \hat{x}_m \cdot \hat{y}_m$ для любого m. Лемма доказана.

Рассмотрим кольцо $\mathrm{R} = \mathbb{Z}_{2^k+1}$. В этом кольце нужные для дальнейшего алгоритма операции сложения, умножения на степень двойки и приведение по модулю $2^k + 1$ выполняются легко, а именно: умножение- это приписывание к числу соответствующее число нулей; остаток от деления на $2^k + 1$ считается исходя из условия $2^k \equiv -1 (mod\, 2^k + 1)$ и достаточно разбить число на блоки по 2^k цифр и вычислить

знакопеременную сумму получившихся чисел.

Будем рассматривать случай $k = 2^l$, $l > 0$.

Лемма 3.

Для всякого $g \in \mathrm{R}$ верно следующее равенство:

$$\sum_{i=0}^{2^l-1} g^i = \prod_{i=0}^{l-1}(1 + g^{2^i})$$

Доказательство.

Докажем индукцией по l. При $l = 1$ утверждение, очевидно, выполнено. Пусть утверждение выполнено для всех чисел, меньших l. Тогда для l:

$$\sum_{i=0}^{2^l-1} g^i = (1 + g) \sum_{i=0}^{2^{l-1}-1} (g^2)^i, \tag{7}$$

По предположению индукции, правую часть (7) можно разложить в произведение:

$$\sum_{i=0}^{2^{l-1}-1} (g^2)^i = \prod_{i=0}^{l-2}(1 + (g^2)^{2^i}) = \prod_{i=1}^{l-1}(1 + g^{2^i});$$

Подставляя это выражение в (7), получаем

$$\sum_{i=0}^{2^l-1} g^i = (1 + g)\prod_{i=1}^{l-1}(1 + g^{2^i}) = \prod_{i=0}^{l-1}(1 + g^{2^i}).$$

Что доказывает лемму.

В кольце R для вычисления преобразования Фурье нужно найти примитивный корень, для этого:

Лемма 4.

Пусть $g \in \mathrm{R}$ и $g^{k/2} + 1 \equiv 0 (mod\, 2^k + 1)$. Тогда для $1 \leqslant p < k$ выполняется:

$$\sum_{i=0}^{k-1} g^{ip} \equiv 0 (mod\, 2^k + 1) \tag{8}$$

Доказательство.

По предыдущей лемме при $k = 2^l$ сумму (8) можно представить в виде

$$\sum_{i=0}^{k-1} g^{ip} = \prod_{i=1}^{l-1}(1 + g^{2^i p})$$

Докажем, что $1 + g^{2^i p} \equiv 0 (mod\, 2^k + 1)$ для некоторого $i : 0 \leqslant i < l$. Пусть $p = 2^f p'$, p'-нечетное число и $0 \leqslant f < l$. Возьмем $i = l - 1 - f$. Тогда $1 + g^{2^i p} = 1 + g^{2^{l-1} p'} =$

$1 + g^{k/2 \cdot p'} \equiv 1 + (-1)^{p'} \equiv 0 \, (mod \, 2^k + 1)$. Т.е. нашли такое i при котором множитель $(1 + g^{2^i p})$ равен нулю, значит, все произведение ноль по $mod \, 2^k + 1$.

Получаем следующую теорему:

Теорема 1.

В кольце $\mathrm{R} = \mathbb{Z}_{2^k+1}$ число 2^2 является примитивным корнем степени k.

Доказательство.

Число 2^2 обладает свойствами:

1. $2^2 \neq 1$;
2. $(2^2)^k = (2^k)^2 = (-1)^2 = 1$;
3. $(2^2)^{k/2} + 1 \equiv 0 (mod \, 2^k + 1)$, значит, можем применить лемму 4 и $\sum_{i=0}^{k-1} g^{ip} = 0$, для $1 \leq p < k$.

по определению получаем, что 2^2 примитивный корень степени k. Аналогично, 2^{2m}-примитивный корень степени k.

Так как k-степень двойки и 2^k+1-нечетное, то эти числа взаимно простые, значит, для них можно выполнять и обратное преобразование Фурье.

Рассмотрим алгоритмы для вычисления прямого и обратного преобразования Фурье.

Сначала введем определение операции $*$, которая будет присутствовать в алгоритме.

Определение 6. Целое число j представляем в двоичной записи и переворачиваем все коэффициенты, другими словами, для $j = 2^{s-1} j_1 + ... + 2 j_{s-1} + j_s$ символом j^* будем обозначать $j^* = 2^{s-1} j_s + ... + 2 j_2 + j_1$.

Алгоритм 1 ***Быстрое преобразование Фурье.***

Дано: $x = (x_0, x_1, ..., x_{2^s-1})$.

Найти: $\hat{x} = (\hat{x}_0, \hat{x}_1, ..., \hat{x}_{2^s-1})$.

g- примитивный корень из 1, степени 2^s, т.е. $g^{2^s} = 1$.

1. Строим матрицу $A = (a_{u,v})$, где $0 \leq u \leq s, 0 \leq v \leq 2^s - 1$ по следующему правилу:

для $u = 0$ положим

$$a_{0,v} = x_{v*}$$

2. для $0 < u \leq s$, при всех v, $0 \leq v < 2^s$: Вычисляем

$$a_{u,v} = a_{u-1,w} + g^{2^{s-u}v} a_{u-1,w+2^{u-1}}, \tag{9}$$

где

$$w = v - i_{s-u+1} \cdot 2^{u-1} = \begin{cases} v, \text{ если } i_{s-u+1} = 0, \\ v - 2^{u-1}, \text{ если } i_{s-u+1} = 1; \end{cases}$$

3. Получаем, что для $0 \leq i < 2^s$

$$a_{s,i} = \hat{x}_i.$$

Утверждение 1 ***Оценка сложности для быстрого преобразования Фурье.***

Количество арифметических операций в алгоритме $5s \cdot (2^s - 1)$.

Доказательство.

В пункте 1 алгоритма- количество арифметических операций $2s \cdot (2^s - 1)$. Так как для каждого v такого, что $0 \leq v \leq 2^s - 1$ " $*$ " занимает $2s$ операций, в связи с тем, что число v в двоичной записи, переворачиваем, а далее каждую цифру нужно умножить на степень двойки и сложить полученные результаты.

В пункте 2 выполянется для каждого $u, v : 0 \leq u \leq s, 0 \leq v \leq 2^s - 1$ по 3 арифметические операции, т.е. всего $2s \cdot (2^s - 1) + 3s \cdot (2^s - 1) = 5s \cdot (2^s - 1)$. Таким образом, сложность алгоритма $5s \cdot (2^s - 1)$.

Для доказательства корректности алгоритма посмотрим, что происходит на каждом шаге, для этого теорема.

Теорема 2.

Для каждых $u, v : 0 \leq u \leq s, 0 \leq v \leq 2^s - 1$; элемент матрицы $a_{u,v}$ представляется в виде:

$$a_{u,v} = \sum_{j^* \equiv v^* (mod\ 2^{s-u})} x_{j*} g^{v \cdot 2^{s-u} \left[\frac{j^*}{2^{s-u}}\right]} \tag{10}$$

Под знаком суммирования подразумевается суммирование по всем j таким что $j^* = v^* (mod\ 2^{s-u})$.

Доказательство.

Докажем индукцией по u для всех $0 \le v \le 2^s - 1$

База: при $u = 0$ имеем:

$$\sum_{j^* \equiv v^* (mod\ 2^s)} x_{j^*} g^{v \cdot 2^s \left[\frac{j^*}{2^s}\right]} = \sum_{j^* \equiv v^* (mod\ 2^s)} x_{j^*} = x_{v^*} = a_{0,v} \text{ -верно.}$$

здесь использовалось равенство $\left[\frac{j*}{2^s}\right] = 0$.

Пусть утверждение верно для всех $a_{u-1,v}$ при $0 \le v \le 2^s - 1$, докажем, что утверждение верно для $a_{u,v}$.

$$a_{u,v} = a_{u-1,w} + g^{2^{s-u}v} a_{u-1,w+2^{u-1}} \tag{11}$$

По предположению индукции

$$a_{u-1,w} = \sum_{j^* \equiv w^* (mod\ 2^{s-u+1})} x_{j^*} g^{w \cdot 2^{s-u+1} \left[\frac{j^*}{2^{s-u+1}}\right]}$$

$$a_{u-1,w+2^{u-1}} = \sum_{j^* \equiv (w+2^{u-1})^* (mod\ 2^{s-u+1})} x_{j^*} g^{(w+2^{u-1}) \cdot 2^{s-u+1} \left[\frac{j^*}{2^{s-u+1}}\right]}$$

при $w = v - i_{s-u+1} \cdot 2^{u-1}$.

Справедливы равенства:

$$\left[\frac{j^*}{2^{s-u}}\right] = 2\left[\frac{j^*}{2^{s-u+1}}\right] + j_{s-u+1};$$

$$(v - i_{s-u+1} \cdot 2^{u-1})^* = v^* - 2^{s-u} \cdot i_{s-u+1}$$

$$(v + (1 - i_{s-u+1}) \cdot 2^{u-1})^* = v^* + 2^{s-u} \cdot (1 - i_{s-u+1})$$

• При $i_{s-u+1} = 0$, имеем

$$a_{u,v} = a_{u-1,v} + g^{2^{s-u} \cdot v} a_{u-1,v+2^{u-1}} =$$

$$\sum_{j^* \equiv v^* (mod\ 2^{s-u+1})} x_{j^*} g^{v \cdot 2^{s-u+1} \left[\frac{j^*}{2^{s-u+1}}\right]} + g^{2^{s-u}v} \sum_{j^* \equiv v^* + 2^{s-u} (mod\ 2^{s-u+1})} x_{j^*} g^{(v+2^{u-1}) \cdot 2^{s-u} \left[\frac{j^*}{2^{s-u}}\right] - 1}$$

Здесь использовалось то, что в первой сумме $j_{s-u+1} = 0$, а во второй $j_{s-u+1} = 1$. Поскольку $2^{s-u}v + (v + 2^{u-1}) \cdot 2^{s-u}\left(\left[\frac{j^*}{2^{s-u}}\right] - 1\right) = v \cdot 2^{s-u}\left[\frac{j^*}{2^{s-u}}\right] + 2^{s-1}\left(\left[\frac{j^*}{2^{s-u}}\right] - 1\right)$, а $\left[\frac{j^*}{2^{s-u}}\right]$ есть нечетное число и $g^{2^s} = 1$, находим

$$a_{u,v} = \sum_{j^* \equiv v^* (mod\ 2^{s-u+1})} x_{j^*} g^{v \cdot 2^{s-u} \left[\frac{j^*}{2^{s-u}}\right]} + \sum_{j^* \equiv v^* + 2^{s-u} (mod\ 2^{s-u+1})} x_{j^*} g^{v \cdot 2^{s-u} \left[\frac{j^*}{2^{s-u}}\right]}$$

Объединение множеств индексов j, входящих в первую и вторую сумму может быть задано условием $j^* \equiv v^*(mod\ 2^{s-u})$. Это доказывает равенство (10) в случае $i_{s-u+1} = 0$.

• При $i_{s-u+1} = 1$, имеем

$$a_{u,v} = a_{u,v-2^{u-1}} + g^{2^{s-u}\cdot v} a_{u-1,v} =$$

$$\sum_{j^* \equiv v^* - 2^{s-u}(mod\ 2^{s-u+1})} x_{j^*} g^{(v-2^{u-1})\cdot 2^{s-u+1}\left[\frac{j^*}{2^{s-u+1}}\right]} + g^{2^{s-u}v} \sum_{j^* \equiv v^*(mod\ 2^{s-u+1})} x_{j^*} g^{v\cdot 2^{s-u}\left(\left[\frac{j^*}{2^{s-u}}\right]-1\right)}$$

Здесь использовалось то, что в первой сумме $j_{s-u+1} = 0$, а во второй $j_{s-u+1} = 1$. Поскольку $2^{s-u}v + v \cdot 2^{s-u}\left(\left[\frac{j^*}{2^{s-u}}\right] - 1\right) = v \cdot 2^{s-u}\left[\frac{j^*}{2^{s-u}}\right]$ и $(v - 2^{u-1}) \cdot 2^{s-u+1}\left[\frac{j^*}{2^{s-u+1}}\right] = v \cdot 2^{s-u}\left[\frac{j^*}{2^{s-u}}\right]$, а $\left[\frac{j^*}{2^{s-u}}\right]$ есть четное число, находим

$$a_{u,v} = \sum_{j^* \equiv v^* - 2^{s-u}(mod\ 2^{s-u+1})} x_{j^*} g^{v\cdot 2^{s-u}\left[\frac{j^*}{2^{s-u}}\right]} + \sum_{j^* \equiv v^*(mod\ 2^{s-u+1})} x_{j^*} g^{v\cdot 2^{s-u}\left[\frac{j^*}{2^{s-u}}\right]}$$

Объединение множеств индексов j, входящих в первую и вторую сумму может быть задано условием $j^* \equiv v^*(mod\ 2^{s-u})$. Это доказывает равенство (10) в случае $i_{s-u+1} = 1$.

Следовательно, утверждение теоремы доказано для всех u.

При $u = s$ получаем

$$a_{u,v} = \sum_{j:j*=v*(mod\ 1)} x_{j^*} g^{j^* v} = \sum_{j=0}^{2^s-1} x_j \cdot g^{jv} = \hat{x}_v.$$

Это доказывает корректность алгоритма.

Алгоритм 2 ***Обратного быстрого преобразования Фурье.***

Дано: $\hat{x} = (\hat{x}_0, \hat{x}_1, ..., \hat{x}_{k-1})$.

Найти: $x = (x_0, x_1, ..., x_{k-1})$.

g^{-1}- примитивный корень из 1, степени 2^s, т.е. $g^{-2^s} = 1$.

1. Строим матрицу $A = (a_{u,v})$, где $0 \le u \le s, 0 \le v \le 2^s - 1$ по следующему правилу:

   $\forall \quad 0 \le v \le 2^s - 1$ положим:

   $$a_{0,v} = \hat{x}_{v*}$$

2. для $0 < u \le s$, при всех v, $0 \le v < 2^s$: Вычисляем

   $$a_{u,v} = a_{u-1,w} + g^{-2^{s-u}v} a_{u-1,w+2^{u-1}},$$

Где

$$w = v - i_{s-u+1} \cdot 2^{u-1} = \begin{cases} v, \text{ если } i_{s-u+1} = 0, \\ v - 2^{u-1}, \text{ если } i_{s-u+1} = 1; \end{cases}$$

3. Получаем, что для $0 \leq i < 2^s$

$$\hat{x}_i = \frac{1}{2^s} a_{s,i}.$$

Видно, что этот алгоритм обратного преобразования напоминает алгоритм для прямого преобразования, с той лишь разницей, что здесь примитивный корень берется, обратный тому, что брали при прямом преобразовании. Т.е. здесь примитивный корень- g^{-1}, все свойства из определения 2 примитивного корня, очевидно, выполняются. Получаем следующую теорему:

Теорема 3.

Положим $a_{0,v} = x_{v*}$ для всех $0 \leq v \leq 2^s - 1$.

Определим для каждых $u, v : 0 \leq u \leq s, 0 \leq v \leq 2^s - 1$

$$a_{u,v} = a_{u-1,w} + g^{-2^{s-u}v} a_{u-1,w+2^{u-1}},$$

$$\text{где } w = v - i_{s-u+1} \cdot 2^{u-1}$$

Тогда $a_{u,v}$ представляется в виде:

$$a_{u,v} = \sum_{j^* \equiv v^* (mod\ 2^{s-u})} x_{j*} g^{-v \cdot 2^{s-u} \left[\frac{j^*}{2^{s-u}}\right]}$$

Доказательство.

Аналогично доказательству теоремы 2. Достаточно заменить в доказательстве теоремы 1 примитивный корень g на g^{-1}.

Утверждение 2 ***Корректность работы алгоритма обратного преобразования.***

В условиях теоремы 3 положим $u = s$, тогда

$$a_{s,v} = \sum_{j : j* = v* (mod\ 1)} \hat{x}_{j*} g^{-j^* v} = \sum_{j=0}^{2^s - 1} \hat{x}_j \cdot g^{-jv} \tag{12}$$

По следствию 1 правая часть равенства (12) есть обратное преобразование Фурье, значит:

$$\sum_{j=0}^{2^s-1} \hat{x}_j \cdot g^{-jv} = x_v \cdot 2^s, \text{ т.е. } \frac{1}{2^s} a_{u,v} = x_v.$$

Кольцо, в котором будем проводить операции и примитивный корень, использующийся для прямого преобразования Фурье и обратного, определен ранее. Теперь опишем, как работает алгоритм, придуманный Шенхаге и Штрассеном в 1971 году.

В этом алгоритме умножения целых чисел идея заключается в предположении, что числа $a, b < 2^n$, тогда выбирается $m \in \mathbb{N}, m \geqslant 4$ такое что $2n \vdots 2^m$ и $k \geqslant 4n/2^m + m$ при этом $2^m \mid 2k$.

Возьмем $k = n$, тогда очевидно, выполняются все требования.

Пусть $k = 2^{2s}$ или 2^{2s+1}, положим $h = 2^s, l = k/h$. И будем представлять числа в l-ичной записи, т.е. $a = \sum_{i=0}^{h-1} x_i 2^{il}, b = \sum_{i=0}^{h-1} y_i 2^{il}$, где $0 \leqslant x_i, y_i < 2^l$. Таким образом, мы разбили числа a и b на h блоков длины l и для их умножения будем использовать следующий алгоритм

Алгоритм 3 ***Умножения целых чисел.***

Дано: числа $a, b \in \mathbb{Z}$, $0 \leq a, b < 2^k$.

Найти: $ab \,(mod\, 2^k + 1)$.

1. Определим наборы $x = (x_0, x_1, ..., x_{h-1})$,$y = (y_0, y_1, ..., y_{h-1})$, где

$$a = \sum_{i=0}^{h-1} x_i 2^{il}, b = \sum_{i=0}^{h-1} y_i 2^{il}$$

   (a_i, b_i- блоки, составленные из l разрядов чисел a, b соответственно)

2. С помощью быстрого преобразования Фурье, вычисляем $\hat{x} = (\hat{x}_0, \hat{x}_1, ..., \hat{x}_{h-1})$, $\hat{y} = (\hat{y}_0, \hat{y}_1, ..., \hat{y}_{h-1})$ взяв в качестве g примитивный корень из 1 степени $2^{2l/b}$.

3. Вычисляем $\hat{x} \cdot \hat{y} = (\hat{x}_0\hat{y}_0, \hat{x}_1\hat{y}_1, ..., \hat{x}_{h-1}\hat{y}_{h-1})(mod\, 2^{2l}+1)$. Притом, для вычисления каждого произведения рекурсивно применяем этот алгоритм умножения, до того момента, пока числа не будут маленькими.

4. Вычисляем обратное преобразование Фурье набора $\hat{x} \cdot \hat{y}$, т.е. находим такой набор $z = (z_0, z_1, ..., z_{h-1})$, z_i считаем числами по $mod\, 2^{2l}$, и такие, что $\hat{z}_i = \hat{x}_i \cdot \hat{y}_i$, тогда $\hat{z} = \hat{x} \cdot \hat{y}$.

$$z_i = \frac{1}{2^s} \sum_{j=0}^{2^s-1} \hat{z}_j g^{-ij}$$

5. $z = \sum_{i=0}^{h-1} z_i 2^{il}$

Произведение чисел a и b мы вычислили по $mod\, 2^k + 1$, тем самым мы нашли не точное их произведение. Для точного, надо изначальную длину чисел k сделать в два раза больше, т.е. $2k$ дописав нужное количество нулей и умножить тем же алгоритмом. Тогда мы найдем их произведение по $mod(2^{2k} + 1)$, а так как сами числа меньше 2^k, то их произведение меньше 2^{2k}, найдем точное значение произведения.

Утверждение 3

Полученное число z будет произведением чисел a и $b(mod\, 2^k + 1)$.

Доказательство.

Разбиение в пункте 1 чисел на блоки, другими словами, можно записать в виде представления их многочленами. Т.е. $f(T) = \sum_{i=0}^{h-1} x_i T^i, g(T) = \sum_{i=0}^{h-1} y_i T^i$, тогда $a = f(2^l), b = g(2^l)$ и их произведение $ab = f(2^l) \cdot g(2^l)$, обозначим $c(x) = \sum_{m=0}^{h-1} c_m x^m = f(x)g(x)$. Таким образом, нам надо найти коэффициенты полученного многочлена $c(x)$. Коэффициенты $c_m = \sum_{i+j \equiv m(mod\, h)} x_i y_j$, по определению 4 получаем, что c_m являются светкой x_i, y_j. Следовательно, эту сумму можно найти по лемме 2 через преобразование Фурье, что мы и сделали.

Утверждение 4 ***Оценка сложности.***

В первом пункте алгоритма происходит разделение на блоки, что занимает $O(h) = O(2^s)$ арифметических операций.

В пункте 2 алгоритма количество операций по утверждению 1 равно $5s \cdot (2^s - 1)$.

В пункте 3 по лемме 5 количество операций $O(k \cdot log\, k \cdot loglog\, k)$.

Пункт 4, аналогичен п.2 и в нем количество операций $5s \cdot (2^s - 1)$.

В пункте 5- $2h = 2^{s+1}$ арифметических операций.

Всего получаем количество операций $O(k \cdot log\, k \cdot loglog\, k)$.

Лемма 5.

Количество операций в пункте 3 алгоритма умножения целых чисел равно $k \cdot log\, k \cdot loglog\, k$.

Доказательство.

Пусть $M(k)$- количество операций для перемножения двух чисел длины $log\, k$. В алгоритме используется рекурсия, то есть нужно применить алгоритм для h чисел,

каждое длины $2l$. Соответственно, получаем оценку

$$M(k) \leqslant ck \cdot log\, k + hM(2l).$$

Обозначим $M'(k) = M(k)/k$, тогда

$$M'(k) \leqslant c \cdot log\, k + hM'(2l),$$

т.к. $l \leqslant 2\sqrt{k}$, то $M'(k) \leqslant c \cdot log\, k + hM'(4\sqrt{k})$.

Докажем по индукции, что в этом случае: $M'(k) \leqslant c'k \cdot log\, k \cdot loglog\, k$, при подходящем выборе c'.

Пусть утверждение верно для $M'(4\sqrt{k})$, докажем утверждение для $M'(k)$.

$$M'(k) \leqslant c \cdot log\, k + 4c'\, log(2 + \frac{1}{2} log\, k) + c'\, log\, k \cdot log(2 + \frac{1}{2} log\, k)$$

Для больших k выполянется $(2 + \frac{1}{2} log\, k) \leqslant \frac{2}{3} log\, k$, отсюда

$$M`(k) \leqslant c \cdot log\, k + 4c'\, log \frac{2}{3} + 4c'\, loglog\, k + c' log \frac{2}{3} log\, k + c' log\, k \cdot loglog\, k$$

Для больших чисел первые три слагаемых не больше абсолютной величины четвертого, таким образом, $M'(k) \leqslant c'k \cdot log\, k \cdot loglog\, k$, а значит, $M(k) \leqslant c \cdot k \cdot log\, k \cdot loglog\, k$.

В настоящее время уже есть более быстрые алгоритмы умножения целых чисел. Так М. Фюрер в 2007 году придумал алгоритм со сложностью $k \cdot log\, k \cdot 2^{O(log^* k)}$, где $log^*(k)$- называется итерированным логарифмом и

$$log^*(k) = min\{n \geqslant 0 | log^{(n)} k \leqslant 1\} \text{ где } log^{(0)} k = k, log^{(n)} k = log\, log^{(n-1)}(k).$$

Известно, что множитель $2^{O(log^* k)}$ растет медленнее любой итерации логарифмов, но все же является неограниченно возрастающей функцией.

В своей работе Фюрер вводит кольцо многочленов над комплексными числами $\mathbb{C}[x]/(x^{2^k} + 1)$ и производит уже над многочленами преобразование Фурье. Основной сложностью при данном выборе кольца является нахождение примитивного элемента. Для этого Фюрер в своей работе использует сведение задачи к подзадачам меньшего размера, а для них он выбирает примитивный корень, являющийся степенью двойки. Тогда умножение и деление на примитивный корень может быть вычислено с помощью сдвига.

Но умножение чисел с помощью этого алгоритма на вычислительной машине представляет сложность. Так как комплексные числа нельзя записать в памяти целиком, а округление приводит к потерям точности ответа.

И в 2008 году А. Де, Ш. Саха, П. Курур и Р. Сантхариши построили похожий алгоритм, основанный уже на модульной, а не комплексной арифметике, достигнув при этом такого же времени работы. Они рассматривают кольцо $\mathrm{R} = \mathbb{Z}[x]/(p^c, x^m+1)$ для некоторого m, константы c и простого p. Элементы в нем так же есть многочлены. Однако в нем не всегда существует примитивный корень, поэтому еще отдельной задачей является выбор параметров кольца.

Список литературы

[1] Ахо А., Хопкрофт Дж., Ульман Дж. Построение и анализ вычислительных алгоритмов

[2] О.Н.Герман, Ю.В.Нестеренко Теоретико-числовые методы в криптографии

[3] Кибернетический сборник Новая серия выпуск 10 под ред. А. А. Ляпунова изд. Мир, Москва 1973

[4] A. Schonhage and V. Strassen, «Schnelle Multiplikation grosser Zahlen», Computing 7 (1971), pp. 281—292.

[5] Furer, M. (2007). «Faster Integer Multiplication» in Proceedings of the thirty-ninth annual ACM symposium on Theory of computing, June 11-13, 2007, San Diego, California,USA

[6] Anindya De, Piyush P Kurur, Chandan Saha, Ramprasad Saptharishi. «Fast Integer Multiplication Using Modular Arithmetic». Symposium on Theory of Computation (STOC) 2008.