Лекция 2

Алгоритм Шенхаге-Штрассена для умножения чисел

(Конспект: А. Бережной)

Пусть нужно перемножить два числа, a_1 и a_2: каждое длиной n бит. Обычный алгоритм умножения “в столбик” имеет сложность $O(n^2)$. Посмотрим, как это можно улучшить.

2.1 Простой алгоритм.

Во-первых, построим простой рекурсивный алгоритм, разделяя числа на “половинки” и рекурсивно перемножая эти “половинки”. Считаем, что $n = 2^k$. Тогда

$$a_1 = u_1 + u_2 2^{k-1} \quad ; \quad a_2 = v_1 + v_2 2^{k-1}$$
$$\Downarrow$$
$$a_1 a_2 = u_1 v_1 + 2^{k-1}(u_1 v_2 + u_2 v_1) + 2^k u_2 v_2.$$

Итого 4 умножения (умножение на степень двойки — это не умножение, а сдвиг). Но пусть:

$$x = u_1 v_1;\ y = (u_1 + u - 2)(v_1 + v_2);\ z = u_2 v_2.$$

Тогда:

$$a_1 a_2 = x + 2^{k-1}(y - x - z) + 2^k z$$

Это уже только 3 умножения. Теперь посчитаем сложность алгоритма. Очевидно, $T(n) = 3T(\frac{n}{2}) + cn$. Отсюда, аналогично теореме 1.2 из первой лекции, имеем

$$T(n) = O(n^{log_2 3}).$$

Это лучше, чем $O(n^2)$, но есть еще более быстрый способ. Для него понадобится понятие *Дискретного Преобразования Фурье (ДПФ).*

2.2 Дискретное преобразование Фурье и его вычисление.

Определение 2.1. Пусть ω — первообразный корень степени b. Рассмотрим какой-нибудь вектор $(a_{b-1}, ..., a_0)$. Его ДПФ — это вектор, состоящий из значений полинома $a_{b-1}x^{b-1} + \ldots + a_1x + a_0$ в точках $1, \omega, \omega^2, ...$.

Быстрое вычисление ДПФ. Пусть $f(x)$ — это тот самый полином, вычисляющий ДПФ. По теореме Безу, $f(x) \mod (x - \omega^i) = f(\omega^i)$. Все, что нам нужно — это быстро находить остатки. Построим двоичное дерево; его узлы будут помечены полиномами. Листья дерева помечены полиномами $x - \omega^i$. Полином в любом узле является произведением полиномов в потомках. Упорядочим листья так, чтобы каждый узел был помечен полиномом вида $x^m - \omega^k$. Это делается так: в качестве k-го листа берем полином $x - \omega^{rev(k)}$, где rev – операция переворачивания битов числа; например, если у нас всего 32 листа (т.е. 5 битов), то $rev(2) = 8$: в двоичной системе, $rev(00010) = 01000$. Очевидно, что $x \vdots 2 => rev_n(\frac{x}{2}) = 2rev_n(x)$. Пользуясь этим, можно доказать (индукцией), что дерево получилось такое, как предполагалось. Легко проверить, что

$$\sum_{i=0}^{2t-1} a_i x^i \mod (x^t - \omega^s) = \sum_{j=0}^{t-1} (a_j + \omega^s a_{j+t}) x^j.$$

Так мы можем быстро вычислить остаток любого полинома по модулю $x^t - w^s$. Спускаясь по дереву, мы будем заменять имеющийся полином на остаток, пока не спустимся до листьев. Так мы получим ДПФ, ведь

$$f(x) \mod (x - \omega^i) = f(x) \mod q_1 \mod q_2 \mod \ldots \mod x - \omega^i,$$

где $q_1 \vdots q_2 \vdots \ldots \vdots x - \omega^i$.

2.3 Умножение при помощи ДПФ.

Вернемся к умножению. Пусть опять $n = 2^k$. На этот раз разобьем оба числа не на два блока, а на b: каждый длиной l бит, причем так, что $b \approx l$, т.е., либо $b = l$, либо $b = \frac{l}{2}$ і (это зависит от четности k, ибо $b \times l = n$). Тогда

$$\begin{aligned} a_1 &= u_0 + u_1 2^l + ... + u_{b-1} 2^{l(b-1)}, \\ a_2 &= v_0 + v_1 2^l + ... + v_{b-1} 2^{l(b-1)}, \\ a_1 \times a_2 &= w_0 + w_1 2^l + ... + w_{2b-2} 2^{l(2b-2)}, \end{aligned}$$

где

$$w_i = y_i + y_{b+i}, \; y_i = \sum_{j=0}^{b-1} u_j v_{i-j} \text{ (а } y_{b+i} = \sum_{j=0}^{b-1} u_j v_{b+i-j})$$

(если какой-то блок под суммой не существует, считаем его нулем).

Вычислять w_i будем так: сначала найдем

$$w_i' = w_i \mod b \quad \text{и} \quad w_i'' = w_i \mod (2^{2l} + 1).$$

После этого, очевидно, w_i находится по формуле:

$$w_i = w_i'(2^{2l} + 1) - w_i'' 2^{2l}.$$

Сначала получим w_i'. Построим два числа следующим образом. Эти числа состоят из b блоков длиной $3 \log b$ каждый. Блок, занимающий в 1-м числе i-ю позицию (справа) состоит из $2 \log b$ нулей и числа u_i. Во втором числе то же самое, но вместо u_i всюду v_i. Перемножить эти два числа можно нашим первым простым алгоритмом за $O((3b \log b)^{\log 3}) = O(b^2) = O(n)$; при этом мы получим число, состоящее из блоков $y_i \mod b$, а стало быть и w_i'.

Как найти w_i''? Тут поможет ДПФ. У нас есть два вектора: $(u_{b-1}\psi^{b-1}, ..., u_0\psi)$ и $(v_{b-1}\psi^{b-1}, ..., v_0\psi)$, где $\psi^2 = \omega$ (в нашем случае $\psi = 2^{2l/b}$; легко проверить, что ψ^2 — действительно первообразный корень b-ой степени в $\mathbb{Z}_{2^{2l}+1}$). Мы вычислим от них ДПФ и почленно перемножим. Утверждается, что результатом будет ДПФ от вектора $(w_{b-1}\psi^{b-1}, ..., w_0\psi)$. Это утверждение называется *Теоремой об отрицательно обернутой свертке.* Докажем ее:

$$(\text{ДПФ}(w_i\psi^i))_j = \sum_i \omega^{ij} w_i \psi^i = \sum_s \omega^{sj} (\sum_m u_m v_{s-m} - \sum_m u_m v_{b+s-m}) \psi^s,$$

$$(\sum_i \omega^{ij} u_i \psi^i)(\sum_k \omega^{kj} v_i \psi^k) = \sum (\omega^{(i+k)j} \psi^{i+k} \sum_m u_m v_{i+k-m}).$$

Правые части равны: $(i+j)$ во втором равенстве играют роль s в первом. Осталось посчитать сложность. Докажем сначала, что ДПФ вычисляется за время $O(n \log n)$. Дерево имеет высоту $\log b$. Пусть на уровне вычисляется s остатков. На каждый остаток выполняется $\frac{b}{s}$ операций: каждая сложностью $2l$. Итого $O(b \log b 2l) = O(n \log n)$. Теперь общее время на вычисление свертки:

$$T(bl) = bT(2l) + cn \log n.$$

Остается решить это рекуррентное уравнение ...

$$T'(s) = \frac{T(s)}{s},$$

$$T'(n) = 2T'(2l) + c \log n,$$

$$T'(n) = 2T'(2\sqrt{n}) + c \log n,$$

$$T''(s) = T'(2^s),$$

$$T''(\log n) = 2T''(\frac{\log n + 1)}{2}) + c \log n,$$

$$T''(s) = 2T''(\frac{s}{2} + 1) + cs.$$

Итого,

$$T(n) = n \log n \log \log n.$$