2.4. Алгоритм Тоома для умножения чисел

Здесь мы рассмотрим еще более быстрый алгоритм для умножения чисел, который предложил А. Л. Тоом [15]. Нам потребуется следующий известный факт о многочленах.

Утверждение (интерполяционная формула Лагранжа). *Пусть* $P(x) = c_n x^n + c_{n-1}x^{n-1} + \ldots + c_1 x + c_0$ — *произвольный полином степени* n, *значения которого* $P(d_m)$ *известны в* $n+1$ *различных точках* $d_1, d_2, \ldots, d_{n+1}$. *Тогда существуют такие константы* α_{qm}, *зависящие только от* $d_1, d_2, \ldots, d_{n+1}$, *что*

$$c_q = \sum_{m=1}^{n+1} \alpha_{qm} P(d_m), \quad q = 0, 1, \ldots, n. \tag{2.4}$$

При этом, если все d_m *рациональны, то и все* α_{qm} *рациональны.*

Теорема 2.6. *Для любого фиксированного* $\varepsilon > 0$ *выполняется* $M(n) = O(n^{1+\varepsilon})$, *где* $M(n)$ — *минимальная битовая сложность умножения двух* n-*разрядных двоичных чисел.*

Доказательство. Зафиксируем натуральное $k \geqslant 2$ и рассмотрим следующий алгоритм Тоома [15] для умножения n-разрядных двоичных чисел A и B. Если $k^{m-1} < n \leqslant k^m$, то увеличим разрядность до k^m, приписывая спереди нули. Для $n = k^m$ поступаем следующим образом. Режем A и B на k кусков длины k^{m-1}. Пусть $A = (A_{k-1}A_{k-2}\ldots A_1A_0)_2$ и $B = (B_{k-1}B_{k-2}\ldots B_1B_0)_2$. Рассмотрим многочлены $f(x) = A_{k-1}x^{k-1} + A_{k-2}x^{k-2} + \ldots + A_1x + A_0$ и $g(x) = B_{k-1}x^{k-1} + B_{k-2}x^{k-2} + \ldots + B_1x + B_0$. Тогда $A = f(2^{k^{m-1}})$, $B = g(2^{k^{m-1}})$ и искомое $C = A \cdot B = f(2^{k^{m-1}}) \cdot g(2^{k^{m-1}}) = h(2^{k^{m-1}})$, где $h(x) = f(x) \cdot g(x)$. Заметим, что $h(x)$ — многочлен степени $2k-2$. Алгоритм состоит из следующих шагов.

1. Вычисляем $f(d_m)$ и $g(d_m)$, где $d_1, d_2, \ldots, d_{2k-1}$ —любые фиксированные целые точки (например, $d_m = 0, \pm 1, \pm 2, \ldots, \pm(k-1)$).

2. Вычисляем $h(d_m) = f(d_m)g(d_m)$ тем же алгоритмом для $n = k^{m-1}$ (мы уточним это ниже).

3. По формуле (2.4) вычисляем коэффициенты c_q ($q = 0, 1, \ldots, 2k-2$) многочлена $h(x)$.

4. Вычисляем $h(2^{k^{m-1}}) = C = AB$.

Оценим теперь сложность каждого шага. Отметим, что $k^m = n$, $k^{m-1} = \frac{n}{k}$ и k — константа.

Шаг 1. На этом шаге вычисляем $f(d_m)$ и $g(d_m)$ непосредственно по формулам многочленов, выполняя все операции "в столбик". При этом, так как все d_m — константы и k — константа, вычисление всех d_m^l $(m = 1, 2, \ldots, 2k-1; \quad l = 2, 3, \ldots, k-1)$ требует константного числа битовых операций и длины всех получаемых чисел ограничены константой (зависящей от k, но не зависящей от n). Поэтому вычисление всех одночленов $A_l d_m^l$ требует $O(n)$ битовых операций и длины получаемых чисел не превосходят $\frac{n}{k} + const$. Аналогично для $B_l d_m^l$. Складывая эти одночлены (k — константа), получаем, что вычисление всех значений $f(d_m)$ и $g(d_m)$ требует $O(n)$ битовых операций и длина всех этих значений не превосходит $\frac{n}{k} + const$.

Шаг 2. На этом шаге нам надо $2k-1$ раз перемножить числа длины не более $\frac{n}{k} + const$. Пусть C и D — 2 таких числа, и $C = (C_1 C_0)_2$, $D = (D_1 D_0)_2$, где длина чисел C_0 и D_0 равна $\frac{n}{k}$. Тогда $C \cdot D = (C_1 \cdot 2^{\frac{n}{k}} + C_0) \cdot (D_1 \cdot 2^{\frac{n}{k}} + D_0) = C_1 D_1 \cdot 2^{\frac{2n}{k}} + (C_1 D_0 + C_0 D_1) \cdot 2^{\frac{n}{k}} + C_0 D_0$. Будем вычислять $C_1 D_1$, $C_1 D_0$, $C_0 D_1$ "в столбик", а $C_0 D_0$ рекурсивно тем же алгоритмом, если длина C_0 и D_0, равная k^{m-1}, больше 1. Если же $k^{m-1} = 1$, то $C_0 D_0$ также вычисляем "в столбик". Пусть $L_T(n)$ — битовая сложность алгоритма Тоома (в худшем случае) для умножения чисел длины n. Тогда число операций на шаге 2 не превосходит $(2k-1)L_T(\frac{n}{k}) + O(n)$, и получающиеся числа имеют длину $O(n)$.

Шаг 3. Так как все d_m — целые, то все α_{qm} в формуле (2.4) рациональные. Пусть β — их общий знаменатель и $\alpha_{qm} = \frac{\beta_{qm}}{\beta}$. Тогда все β_{qm} — целые и $c_q = \frac{1}{\beta} \sum_{m=1}^{2k-1} \beta_{qm} h(d_m)$. Так как k — константа, все β_{qm} — константы и длина всех чисел $h(d_m)$ есть $O(n)$, то для вычисления всех сумм $\sum_{m=1}^{2k-1} \beta_{qm} h(d_m), q = 0, 1, \ldots, 2k-1$, требуется $O(n)$ битовых операций и при этом получаются числа длины $O(n)$. Так как β — константа, то вычисление всех c_q (которые заведомо должны быть целыми, как коэффициенты многочлена $h(x) = f(x)g(x)$) требует $O(n)$ битовых операций (делим "в столбик"), и все c_q имеют длину $O(n)$.

Шаг 4. Вычисление $h(2^{k^{m-1}})$ сводится к сложению чисел c_q, сдвинутых влево не более, чем на n разрядов. Так как чисел c_q константное количество, то вычисление $h(2^{k^{m-1}}) = C = AB$ требует $O(n)$ битовых операций.

Для общего числа $L_T(n)$ битовых операций в описанном алгоритме (при $n = k^m$) имеем

$$L_T(n) \leqslant (2k-1)L_T(\frac{n}{k}) + O(n).$$

Тогда по теореме 2.4 о рекуррентном неравенстве для всех n получаем

$$L_T(n) = O(n^{\log_k(2k-1)}).$$

С ростом k имеем

$$\log_k(2k-1) = 1 + \log_k(2 - \frac{1}{k}) \longrightarrow 1.$$

Поэтому для любого $\epsilon > 0$ можно выбрать k так, что $\log_k(2k-1) < 1+\epsilon$ и $L_T(n) = O(n^{1+\epsilon})$. Теорема доказана.

Замечание. Еще более быстрым является алгоритм умножения чисел Шенхаге и Штрассена, битовая сложность которого равна $O(n \log n \log\log n)$ [16].