

# Лекция 3 от 16.09.2016. Алгоритм Карацубы, алгоритм Штрассена

## Перемножение 2 длинных чисел с помощью FFT

Пусть  $x = \overline{x_1 x_2 \dots x_n}$  и  $y = \overline{y_1 y_2 \dots y_n}$ . Распишем их умножение в столбик:

$$+ \frac{\begin{array}{c} x_1 x_2 \dots x_n \\ y_1 y_2 \dots y_n \\ z_{11} z_{12} \dots z_{1n} \\ z_{21} z_{22} \dots z_{2n} \\ \vdots \\ z_{n1} z_{n2} \dots z_{nn} \end{array}}{z_1 z_2 \dots z_n}$$

Понятно, что наивное умножение 2 длинных чисел будет иметь сложность  $\mathcal{O}(n^2)$ .

Давайте научимся перемножать 2 числа быстрым преобразованием Фурье за  $\mathcal{O}(n \log n)$ .

Пусть  $a = \overline{a_{n-1} \dots a_0}, b = \overline{b_{n-1} \dots b_0}$ .

Тогда введём многочлены  $f(x) = \sum_{i=0}^{n-1} a_i x^i, g(x) = \sum_{i=0}^{n-1} b_i x^i$ .

За  $\mathcal{O}(n \log n)$  мы можем найти  $h(x) = (f(x) \cdot g(x)) = \sum_{i=0}^{2n-2} c_i x^i$ .

После этого надо аккуратно провести переносы разрядов таким образом и после этого развернуть полученное число, отбросив ненужные нули в начале:

---

**Algorithm 1** Умножение 2 длинных чисел.

```

1: function УМНОЖЕНИЕ_2_ДЛИННЫХ_ЧИСЕЛ( $h(x)$ )  $\triangleright h(x)$  — перемножение 2 многочленов
    $f(x)$  и  $g(x)$ .
2:    $carry \leftarrow 0$ 
3:   for  $i \leftarrow 0$  to  $2n - 1$  do
4:      $h_i \leftarrow h_i + carry$ 
5:      $carry \leftarrow \lfloor \frac{h_i}{10} \rfloor$ 
6:      $h_i \leftarrow h_i \bmod 10$ 

```

Но этот метод плохо применим на практике из-за того, что быстрое преобразование Фурье имеет очень большую константу.

## Алгоритм Карацубы

Какое-то время человечество не знало алгоритмов перемножения быстрее, чем за  $\mathcal{O}(n^2)$ . А.Н. Колмогоров считал, что это вообще невозможно. В один момент собрались математики на мехмате МГУ и решили доказать, что это невозможно. Но один из аспирантов (Анатолий Алексеевич Карацуба) Колмогорова пришёл и сказал, что у него получилось сделать это быстрее. Давайте посмотрим, как:

Будем считать, что  $n = 2^k$  (если это не так, дополним нулями, сложность вырастет лишь в константу раз).

Для начала просто попробуем воспользоваться стратегией «Разделяй и властвуй». Разобьём числа в разрядной записи пополам. Тогда

$$\begin{aligned} & \times \begin{cases} x = 10^{n/2}a + b \\ y = 10^{n/2}c + d \end{cases} \\ & \quad \downarrow \\ & xy = 10^na c + 10^{n/2}(ad + bc) + bd \end{aligned}$$

Как видно, получается 4 умножения чисел размера  $\frac{n}{2}$ . Так как сложение имеет сложность  $\Theta(n)$ , то

$$T(n) = 4T\left(\frac{n}{2}\right) + \Theta(n)$$

Чему равно  $T(n)$ ? Если посмотреть на дерево исходов или воспользоваться индукцией, то получим, что  $T(n) = \mathcal{O}(n^2)$ , что, конечно, неэффективно.

Анатолий Алексеевич проявил недюжие способности и предложил следующее:

Разложим  $(a + b)(c + d)$ :

$$(a + b)(c + d) = ac + (ad + bc) + bd \implies ad + bc = (a + b)(c + d) - ac - bd$$

Подставим это в начальное выражение для  $xy$ :

$$xy = 10^na c + 10^{n/2}((a + b)(c + d) - ac - bd) + bd$$

Отсюда видно, что достаточно посчитать три числа размера  $\frac{n}{2}$ :  $(a + b)(c + d)$ ,  $ac$  и  $bd$ . Тогда:

$$T(n) = 3T\left(\frac{n}{2}\right) + \Theta(n)$$

Докажем, что  $T(n) = \mathcal{O}(n^{\log_2 3})$ .

Рассмотрим дерево исходов: в каждой вершине дерева мы выполняем не более  $Cm$  действий, где  $C$  —какая-то фиксированная константа, а  $m$  — размер числа на данном шаге, поэтому

$$T(n) \leq Cn \left(1 + \frac{3}{2} + \dots + \frac{3^{\log_2 n}}{2^{\log_2 n}}\right), \text{ так как на каждом шаге мы запускаемся 3 раза от задачи в 2 раза}$$

$$\text{Откуда } T(n) \leq Cn \cdot \frac{\frac{3^{\log_2 n} - 1}{3/2 - 1}}{1/2} = 2Cn^{\log_2 3} = \mathcal{O}(n^{\log_2 3}) \approx \mathcal{O}(n^{1.5849})$$

Полученный алгоритм называется алгоритмом Карацубы.

## Перемножение матриц. Алгоритм Штрассена

После идеи А.А. Карацубы, появились многие алгоритмы, использующие ту же идею. Одним из этих алгоритмов является алгоритм Штрассена. Будем считать, что  $n = 2^k$  снова (оставляем читателю самим подумать, как дополнить матрицы  $m \times t, t \times u$ , чтобы потом легко восстановить ответ)

Пусть у нас есть квадратные матрицы

$$A = \begin{pmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \dots & a_{nn} \end{pmatrix} \text{ и } B = \begin{pmatrix} b_{11} & b_{12} & \dots & b_{1n} \\ b_{21} & b_{22} & \dots & b_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ b_{n1} & b_{n2} & \dots & b_{nn} \end{pmatrix}$$

Сколько операций нужно для умножения матриц? Умножим их по определению. Матрицу  $C = AB$  заполним следующим образом:

$$c_{ij} = \sum_{k=1}^n a_{ik} b_{kj}$$

Всего в матрице  $n^2$  элементов. На получение каждого элемента уходит  $\mathcal{O}(n)$  операций (умножение за константное время и сложение  $n$  элементов). Тогда умножение требует  $n^2 \mathcal{O}(n) = \mathcal{O}(n^3)$  операций.

Попробуем применить аналогичную стратегию «Разделяй и властвуй». Представим матрицы  $A$  и  $B$  в виде:

$$A = \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix} \text{ и } B = \begin{pmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{pmatrix}$$

где каждая матрица имеет размер  $\frac{n}{2}$ . Тогда матрица  $C$  будет иметь вид:

$$C = \begin{pmatrix} A_{11}B_{11} + A_{12}B_{21} & A_{11}B_{12} + A_{12}B_{22} \\ A_{21}B_{11} + A_{22}B_{21} & A_{21}B_{12} + A_{22}B_{22} \end{pmatrix}$$

Как видно, получаем 8 перемножений матриц порядка  $\frac{n}{2}$ . Тогда

$$T(n) = 8T\left(\frac{n}{2}\right) + \mathcal{O}(n^2)$$

По индукции получаем, что  $T(n) = \mathcal{O}(n^{\log_2 8}) = \mathcal{O}(n^3)$ .

Можно ли уменьшить число умножений до 7? Алгоритм Штрассена утверждает, что можно. Он предлагает ввести следующие матрицы (даже не спрашивайте, как до них дошли):

$$\begin{cases} M_1 = (A_{11} + A_{22})(B_{11} + B_{22}); \\ M_2 = (A_{21} + A_{22})B_{11}; \\ M_3 = A_{11}(B_{12} - B_{22}); \\ M_4 = A_{22}(B_{21} + B_{11}); \\ M_5 = (A_{11} + A_{12})B_{22}; \\ M_6 = (A_{21} - A_{11})(B_{11} + B_{12}); \\ M_7 = (A_{12} - A_{22})(B_{21} + B_{22}); \end{cases}$$

Тогда

$$\begin{cases} C_1 = M_1 + M_4 - M_5 + M_7; \\ C_2 = M_3 + M_5; \\ C_3 = M_2 + M_4; \\ C_4 = M_1 - M_2 + M_5 + M_6; \end{cases}$$

Можно проверить что всё верно (оставим это как ~~наказание~~ упражнение читателю). Сложность алгоритма:

$$T(n) = 7T\left(\frac{n}{2}\right) + \mathcal{O}(n^2) \implies T(n) = \mathcal{O}(n^{\log_2 7}) \approx \mathcal{O}(n^{2.8073})$$

Доказательство времени работы такое же, как и в алгоритме Карацубы.

Также существует модификация алгоритма Штрассена, где используется лишь 15 сложений матриц на каждом шаге, вместо 18 предъявленных выше.

## Эквивалентность асимптотик некоторых алгоритмов

Этот раздел не войдёт в экзамен.

Здесь мы поговорим об обращении и перемножении 2 матриц. Докажем, что асимптотики этих алгоритмов эквивалентны.

**Теорема 1** (Умножение не сложнее обращения). *Если можно обратить матрицу размеров  $n \times n$  за время  $T(n)$ , где  $T(n) = \Omega(n^2)$ , и  $T(3n) = \mathcal{O}(T(n))$  (условие регулярности), то две матрицы размером  $n \times n$  можно перемножить за время  $\mathcal{O}(T(n))$*

*Доказательство.* Пусть  $A$  и  $B$  матрицы одного порядка размера  $n \times n$ . Пусть

$$D = \begin{pmatrix} I_n & A & 0 \\ 0 & I_n & B \\ 0 & 0 & I_n \end{pmatrix}$$

Тогда легко понять, что

$$D^{-1} = \begin{pmatrix} I_n & -A & AB \\ 0 & I_n & -B \\ 0 & 0 & I_n \end{pmatrix}$$

Матрицу  $D$  мы можем построить за  $\Theta(n^2)$ , которое является  $\mathcal{O}(T(n))$ , поэтому с условием регулярности получаем, что  $M(n) = \mathcal{O}(T(n))$ , где  $M(n)$  — асимптотика перемножения 2 матриц.  $\square$

С обратной теоремой предлагаем ознакомиться в книге Кормена или Ахо, Хопкрофта и Ульмана.