Дисциплина

«РАЗРАБОТКА ЭФФЕКТИВНЫХ АЛГОРИТМОВ»

Лекция 17

ОТ ГИПОТЕЗЫ КОЛМОГОРОВА ДО БЫСТРОГО ПРЕОБРАЗОВАНИЯ ФУРЬЕ

АЛГОРИТМЫ УМНОЖЕНИЯ ДЛИННЫХ ЦЕЛЫХ ЧИСЕЛ

Лектор: Михаил Васильевич Ульянов,

muljanov@mail.ru, **8 916 589 94 04**

1. История вопроса

В середине 1950-х А. Н. Колмогоров работал над очерком по истории математики. Его наблюдение — за почти 5000 лет письменной истории математики для умножения чисел в позиционной системе не придумали ничего, кроме умножения в столбик в различных модификациях («русский народный алгоритм умножения» — американский термин). Гипотеза Колмогорова — теоретическая нижняя граница умножения — n^2.

Области применения длинных целых (битовая длина больше регистра процессора) — точные вычисления, криптография и т.д.

Представление длинных целых (в данной лекции) — массив из n элементов, каждый хранит один бит числа — двоичная позиционная система счисления.

Представление, рациональное по временной эффективности — массив, где каждый элемент хранит целое с битовой длиной, равной регистру процессора.

2. Умножение в столбик и модификации

Рассмотрим умножение двух чисел 111 и 101.

Очевидно, что умножение сводится к сложению со сдвигом. Заметим, что трудоемкость умножения зависит от порядка сомножителей! При умножении 111 на 101 необходимо одно сложение, а при умножении 101 на 111 — два!

Приведем запись алгоритма сложения.

```
Sum (L, P, Q, S)
(P, Q - исходные числа - массивы длиной n)
(S - массив результата длины n+1)
  sm ← 0(начальный перенос в следующий разряд)  1
  For i ← n downto 1           1+3*n
    sm ← P[i]+Q[i]+sm          5*l
    S[i] ← sm mod 2            3*l
    sm ← sm div 2              2*l
  end
  S[0] ← sm                    2
End.
```

Очевидно, что трудоемкость линейна — 13*n +4.

Идеи алгоритмов:

1. *Последовательное сложение со сдвигом*

Число со сдвигом складывается (с распространением переноса) с текущим результатом всякий раз, когда бит множителя равен 1. Поскольку сложение имеет очевидную оценку $\Theta(n)$, а в среднем мы имеем $n/2$ единичных бит, то совокупно получаем $\Theta(n^2)$.

2. *Сложение с накоплением переноса*

Первый этап. В результирующем массиве происходит сложение бит без переноса с накоплением за $\Theta(n^2)$. Элементы массива могут быть больше единицы. Например, при умножении 111 на 111 имеем 12321.

Второй этап — приведение результата к двоичному виду за $\Theta(n)$. Это распространение общего переноса. Мультипликативная константа у n^2 меньше, чем у алгоритма с последовательным сложением.

3. Алгоритм Карацубы

Математическое обоснование алгоритма.

Метод декомпозиции позволяет, в целом ряде случаев, получить рекурсивные алгоритмы, достаточно эффективные по асимптотической оценке. Для задачи умножения длинных целых чисел использование этого метода позволило А. А. Карацубе впервые получить в 1962 г. алгоритм умножения двух n битовых чисел, имеющий асимптотическую оценку, лучшую, чем $\Theta\ (n^2)$.

Изложим основные идеи этого алгоритма. Пусть a, b — два n битовых числа в обычном двоичном представлении, т. е. старший бит расположен слева. Количество разрядов n таково, что $n = 2^k$ — это обеспечивает целое деление n пополам без остатка на всех рекурсивных вызовах. Обозначим через nh значение $n/2$. Применим метод декомпозиции с разделением задачи на две равные части на любом уровне рекурсии, и представим числа a, b в виде

$$a = 2^{nh} \cdot a_2 + a_1, \quad b = 2^{nh} \cdot b_2 + b_1,$$

где a_1, a_2, b_1, b_2 — числа, имеющие длину в $nh = n/2$ бит (разрядов), причем a_2, b_2 — старшие nh разрядов чисел a, b. Прямая декомпозиция дает:

$$a \cdot b = 2^n \cdot a_2 \cdot b_2 + 2^{nh} \cdot (a_1 \cdot b_2 + a_2 \cdot b_1) + a_1 \cdot b_1.$$

Мы получаем четыре умножения и по теореме имеем оценку $\Theta\ (n^2)$.

Но, произведение $a \cdot b$ может быть записано в следующем виде, и это есть основная идея, предложенная А. А. Карауцбой

$$a \cdot b = 2^n \cdot a_2 \cdot b_2 + 2^{nh} \cdot \big((a_1 + a_2) \cdot (b_1 + b_2) - (a_1 \cdot b_1 + a_2 \cdot b_2)\big) + a_1 \cdot b_1.$$

Произведения в этой формуле могут быть вычислены рекурсивно, но мы должны обеспечить умножение чисел, имеющих длину nh, это очевидно для чисел a_1, a_2, b_1, b_2 — они получены делением двух n битовых чисел пополам. Однако числа $a_1 + a_2$, и $b_1 + b_2$ могут иметь $nh + 1$ двоичных разрядов. Но, в этом случае, требуется не так много дополнительных операций.

Таким образом, мы свели задачу умножения двух n битовых чисел к умножению **трех** пар чисел, имеющих ровно $nh = n/2$ разрядов.

Вычислительная сложность алгоритма. Получим вычислительную сложность алгоритма Карацубы, т. е. асимптотическую оценку функции трудоемкости, на основе следующих рассуждений. Очевидно, что пять сложений, одно вычитание и операции сдвига на n и $n/2$ разрядов, заданные формулой, требуют не более чем $\Theta(n)$ элементарных операций.

В данном случае мы предполагаем, что числа настолько велики, что они хранятся в массивах, каждый элемент которого содержит один бит числа. Мы останавливаем рекурсию при значении $n = 1$, когда произведение $a \cdot b$ вычисляется элементарно, и требует не более чем фиксированного числа базовых операций, которое мы обозначим через C. Поскольку мы рекурсивно перемножаем три пары чисел половинной длины, то приведенные выше рассуждения позволяют записать рекуррентное соотношение для функции трудоемкости исследуемого алгоритма

$$\begin{cases} f_A(1) = C; \\ f_A(n) = 3 \cdot f_A(n/2) + \Theta(n). \end{cases}$$

Используя основную теорему о рекуррентных соотношениях (Бентли, Хакен, Сакс), мы немедленно получаем оценку

$$f_A(n) = \Theta\left(n^{log_2 3}\right), \quad n^{log_2 3} \approx n^{1,5849}.$$

(«Логарифм серединки больше половинки»! Для функций, выпуклых вверх!)

Это асимптотически лучше умножения «в столбик» с оценкой $\Theta(n^2)$.

В случае если $n \neq 2^k$, мы определяем k из неравенства $2^{k-1} < n \leq 2^k$, т. е. $k = \lceil log_2 n \rceil$, и дополняем числа a, b до k разрядов нулями слева, получая следующую асимптотическую оценку

$$n^{log_2 3} = 3^{log_2 n} \Rightarrow f_A(n) = \Theta\left(3^{\lceil log_2 n \rceil}\right).$$

Существует ли возможность улучшения полученной асимптотической оценки этого алгоритма, связанная с разбиением чисел a, b на большее, чем два, количество слагаемых?

Можете придумать формулы, аналогичные формулам Карацубы для разбиения на три? Прямая декомпозиция приведет к 9 умножениям если вы придумаете реализацию за 6 умножений, то показатель степени будет равен 1.6309, что хуже показателя Карацубы (1.5849). А вот если за 5 умножений, то показатель будет равен 1.4649. Возможно ли сократить число умножений до 5?

Даже, если это возможно, то такая декомпозиция приведет к увеличению мультипликативной константы, скрываемой за Θ оценкой.

4. Алгоритм Штрассена-Шёнхаге

1. Усложняем задачу — вместо умножения чисел переходим к умножению полиномов, коэффициенты которых есть биты чисел!

2. Рассмотрим способы хранения полиномов: хранение в коэффициентах, хранение в значениях. При хранении в значениях при умножении полиномов просто умножаются значения этих полиномов в выбранных точках. Однако переход от одного способа к другому имеет квадратичную сложность.

3. Решение — точки, в которых вычисляются значения — комплексные корни из 1 степени $2n$. А это можно сделать быстро с помощью алгоритма Кули и Тьюки — быстрое преобразование Фурье (FFT). Вообще дискретное преобразование Фурье (DFT) есть вычисление значения полинома в точках комплексных корней из единицы (и одновременно разложение сигнала по отсчетам в тригонометрический ряд Фурье!!!)

Совмещая эти преобразования (обратите внимание, что значения исходных полиномов, соответствующих перемножаемым числам, вычисляются не в n точках, а в $2n$ точках, для обеспечения представления результирующего полинома в значениях) получаем:

$$C = {FFT_{2n}}^{-1}\big(FFT_{2n}(A) \cdot FFT_{2n}(B)\big).$$

Обратное преобразование переводит представление в значениях в представление в коэффициентах и остается только перевести полученный кортеж коэффициентов в представление двоичного числа — распространение переноса. Т.е. мы получаем уже известную ситуацию — при умножении 111 на 111 имеем 12321 и нужно распространить перенос.

Поскольку мы трижды выполняем быстрое преобразование Фурье, то в результате имеем алгоритм с асимптотикой $f_A(n) = \Theta(n \cdot ln\, n \cdot lnln\, n)$:

5. Литература

1. Кормен Т., Лейзерсон Ч., Ривест Р., Штайн К. Алгоритмы: построение и анализ, 2-ое издание: 2005. — 1296 с.
2. Карацуба Е. А. Быстрые алгоритмы и метод БВЕ, 2008.
3. Алексеев В. Б. От метода Карацубы для быстрого умножения чисел к быстрым алгоритмам для дискретных функций // Тр. МИАН. — 1997. — Т. 218. — С. 20–27.
4. Карацуба А., Офман Ю. Умножение многозначных чисел на автоматах // Доклады Академии Наук СССР. — 1962. — Т. 145, № 2.
5. Karacuba A. Berechnungen und die Kompliziertheit von Beziehungen (нем.) // Elektronische Informationsverarbeitung und Kybernetik. — 1975. — Bd. 11.
6. Карацуба А. А. Сложность вычислений // Тр. МИАН. — 1995. — Т. 211. — С. 186–202.