А. А. Разборов

Алгебраическая сложность

МЦНМО

Летняя школа «Современная математика»
Дубна, июль 2010

А. А. Разборов

Алгебраическая сложность

Электронное издание

Москва
Издательство МЦНМО
2016

УДК 510.5
ББК 22.12
Р17

Разборов А. А.
Алгебраическая сложность
Электронное издание
М.: МЦНМО, 2016
31 с.
ISBN 978-5-4439-3032-9

Брошюра написана по материалам курса, прочитанного автором в 2010 г. в Летней школе «Современная математика». В ней рассказывается об основных понятиях теории алгебраической сложности и приводятся её начальные утверждения. Рассматриваются задачи эффективного вычисления полиномов и билинейных форм, матричного умножения и алгебраической теории NP-полноты.

Книга представляет интерес для широкого круга сравнительно подготовленных читателей, интересующихся математикой.

Подготовлено на основе книги: *А. А. Разборов*. Алгебраическая сложность. — М.: МЦНМО, 2016. — ISBN 978-5-4439-1032-1.

Издательство Московского центра
непрерывного математического образования
119002, Москва, Большой Власьевский пер., 11,
тел. (499) 241–08–04.
`http://www.mccme.ru`

ISBN 978-5-4439-3032-9

Теория алгебраической сложности была создана во многом благодаря усилиям немецкого математика Фолькера Штрассена [16], которому принадлежат многие классические теоремы в этой области; ее название связано с тем, что она оперирует с полиномами. В алгебраической сложности, как и во многих других разделах теории сложности, имеется много важных задач, которые очень просто формулируются, но остаются открытыми уже в течение десятилетий. Ниже речь пойдет о классических результатах этой области, которым приблизительно 30—40 лет.

1. Вычисление полиномов от одной переменной

Начнем мы с некоторых примеров. Пусть мы хотим вычислить значение полинома x^{2010}, проще говоря, возвести число x в 2010-ю степень. Можно действовать так: заведем переменную, которую сначала положим равной x, а затем 2009 раз умножим ее на x.

Чему равно время работы такого алгоритма? Имеется следующая простая универсальная формула:

$$\text{время работы алгоритма} = \text{число элементарных операций} \times \\ \times \text{время выполнения одной операции.}$$

Интуитивно понятно, что разные арифметические операции (сложение, умножение и деление) имеют, вообще говоря, разную трудоемкость, поэтому в эту формулу часто вводят неотрицательные веса, приписываемые операциям разного типа. Для простоты мы ограничимся случаем, когда часть весов равны 0 (т. е. операция допускается бесплатно), а остальные равны 1. Помимо этого мы абстрагируемся от такой величины, как время выполнения одной элементарной операции. Эта величина зависит от скорости процессора и от других параметров, не связанных с математикой. Кроме того, в алгебраической сложности нас не будет интересовать конкретное значение числа x. Ясно, что на практике намного проще в 2010-ю степень возвести двойку, чем число из тысячи знаков. Но для нас «x умножить на y» — это одна операция, вне зависимости от того, чему равны x и y. Операции сложения и вычитания мы будем считать бесплатными. Более того, операции с фиксированными числами мы тоже будем считать бесплатными, например, вычисление $1000x$ или даже $\sqrt{2}x$ (конечно, с некоторой погрешностью). Причина этого в том, что вычисление $1000x$ можно заменить

многократным сложением, а для $\sqrt{2}x$ можно использовать сложение и деление на два, которое для компьютера тоже является быстрой операцией. Скажем, вычисление выражения $(2x+3)(yz+1)$ займет два действия: умножение y на z и перемножение посчитанных скобок. Все остальные действия при данном вычислении бесплатны.

Итак, приведенный выше алгоритм, как говорят, имеет сложность 2009 — он использует 2009 умножений. Существует более быстрый алгоритм [12]. Возведем x в квадрат 10 раз. Получим:

$$x, x^2, x^{2^2}, \ldots, x^{2^{10}}.$$

Мы посчитали значение x^m для всех m, не превосходящих 2010, являющихся степенями двойки. Теперь представим 2010 в двоичной записи:

$$2010 = 11111011010_2.$$

Эта запись просто означает, что

$$2010 = 2^{10} + 2^9 + 2^8 + 2^7 + 2^6 + 2^4 + 2^3 + 2^1.$$

Чтобы теперь вычислить x^{2010}, достаточно перемножить:

$$x^{2010} = x^{2^{10}} x^{2^9} x^{2^8} x^{2^7} x^{2^6} x^{2^4} x^{2^3} x^{2^1}.$$

Легко видеть, что мы перемножили x^{2^t} по всем тем t, для которых на $(t+1)$-м справа месте в двоичной записи числа 2010 стоит единица. И без точных подсчетов понятно, что нам понадобилось гораздо меньше, чем 2009 умножений. Кроме того, легко оценить, сколько такому методу понадобится умножений для возведения x в произвольную степень n. Ясно, что это число не превосходит длины двоичной записи n, умноженной на какую-то константу. Поскольку, как известно, длина двоичной записи n примерно равна $\log_2(n)$, мы получили алгоритм для вычисления x^n сложности $O(\log_2(n))$ (по поводу обозначения $O(\cdot)$ см. приложение D). Ускорение по сравнению с первым алгоритмом, который для вычисления x^n использует $n-1$ умножение, просто огромно.

Попробуем теперь посчитать значение другого полинома:

$$1 + x + x^2 + \ldots + x^{n-1}.$$

Предположим ненадолго, что в качестве элементарной операции нам разрешено деление. Вспомним формулу для суммирования первых n членов геометрической прогрессии:

$$1 + x + x^2 + \ldots + x^{n-1} = \sum_{i=0}^{n-1} x^i = \frac{1-x^n}{1-x}.$$

Мы уже знаем, как вычислить x^n за $O(\log_2(n))$ умножений. Для получения окончательного ответа понадобится сделать еще одно деление.

Что будет, если деление все-таки запретить? Сможем ли мы все равно вычислить этот полином за $O(\log_2(n))$ операций, если разрешено только умножение?

Оказывается, что да. Для простоты будем считать, что n — степень двойки, т. е. $n = 2^m$. Вычислим, как и раньше, следующие степени x:

$$x, x^2, x^{2^2}, \ldots, x^{2^{m-1}}.$$

Заметим, что

$$\frac{1-x^{2^m}}{1-x} = \sum_{i=0}^{2^m-1} x^i = (1+x+\ldots+x^{2^{m-1}-1})(1+x^{2^{m-1}}).$$

Отсюда видно, что нам необходимо выписать числа вида $1+x^i$ для i от 1 до 2^{m-1} и перемножить. Это означает, что всего нам понадобится $O(m) = O(\log_2(n))$ умножений; заметим, что эту процедуру при желании можно также представить и в рекурсивном виде.

Мы привели пример полинома, который можно быстро вычислить при помощи умножения и деления, а потом оказалось, что так же быстро его можно вычислить, если использовать только умножение. Оказывается (и это первый нетривиальный результат в алгебраической сложности, который мы здесь упомянем — он получен как раз Штрассеном в [15]), что если у нас есть алгоритм для вычисления некоторого полинома (не обязательно от одной переменной) с делениями, то есть и алгоритм, не использующий делений, причем сложность его лишь незначительно больше сложности исходного алгоритма.

Приведем важный пример, где этот результат используется. Рассмотрим задачу вычисления определителя[1]:

$$\det \begin{pmatrix} x_{11} & x_{12} & \ldots & x_{1n} \\ \ldots\ldots\ldots\ldots\ldots & & & \\ x_{n1} & x_{n2} & \ldots & x_{nn} \end{pmatrix} = \sum_{\sigma \in S_n} \operatorname{sgn}(\sigma) x_{1\sigma(1)} \ldots x_{n\sigma(n)}.$$

Определитель — это полином степени n от n переменных. Если мы просто распишем определитель по этой формуле, то мы получим сумму из $n!$ одночленов. Вычисление всех этих одночленов займет у нас недопустимо много времени. Чтобы понять, как вычислить определитель быстрее, вспомним *метод Гаусса*.

Метод Гаусса приводит любую матрицу к верхнетреугольной, т. е. к матрице, у которой на всех местах ниже диагонали стоят нули. Дей-

[1] Необходимые сведения об определителе матрицы даны в приложении В.

ствует он следующим образом. Пусть изначально нам дана матрица

$$\begin{pmatrix} x_{11} & x_{12} & \dots & x_{1n} \\ x_{21} & x_{22} & \dots & x_{2n} \\ \dots & \dots & \dots & \dots \\ x_{n1} & x_{n2} & \dots & x_{nn} \end{pmatrix}.$$

Рассмотрим первый столбец. Если в нем стоят одни нули, то мы сразу переходим к следующему шагу. Иначе найдется номер $j \in \{1, 2, \dots, n\}$ такой, что $x_{j1} \neq 0$. Для простоты можно считать, что $j = 1$ (если это не так, поменяем местами j-ю и 1-ю строчки в матрице). Теперь вычтем первую строчку матрицы из всех оставшихся так, чтобы первый столбец, за исключением x_{11}, занулился. Для этого произведем сначала $n - 1$ деление:

$$\lambda_2 = \frac{x_{21}}{x_{11}},$$
$$\lambda_3 = \frac{x_{31}}{x_{11}},$$
$$\dots\dots\dots$$
$$\lambda_n = \frac{x_{n1}}{x_{11}}.$$

Затем вычислим новые элементы матрицы по следующей формуле:

$$x'_{ji} = x_{ji} - \lambda_j x_{1i}$$

для всех $j \in \{2, \dots, n\}$, $i \in \{1, \dots, n\}$. В результате у нас получится матрица вида:

$$\begin{pmatrix} x_{11} & x_{12} & \dots & x_{1n} \\ 0 & x'_{22} & \dots & x'_{2n} \\ \dots & \dots & \dots & \dots \\ 0 & x'_{n2} & \dots & x'_{nn} \end{pmatrix}.$$

Мы занулили все элементы матрицы под диагональю в первом столбце. Сделаем то же самое со вторым столбцом, третьим и так далее. В конце концов мы получим верхнетреугольную матрицу

$$\begin{pmatrix} a_{11} & a_{12} & \dots & a_{1(n-1)} & a_{1n} \\ 0 & a_{22} & \dots & a_{2(n-1)} & a_{2n} \\ \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & \dots & 0 & a_{nn} \end{pmatrix}.$$

Определитель такой матрицы легко вычислить (см. приложение В, формулу (3)) — он равен произведению элементов на диагонали:

$$a_{11} a_{22} \dots a_{nn}.$$

Содержание