ЛЕТНЯЯ ШКОЛА
СОВРЕМЕННАЯ МАТЕМАТИКА

А. А. Разборов

Алгебраическая сложность

Летняя школа «Современная математика»
Дубна, июль 2010

А. А. Разборов

Алгебраическая сложность

Электронное издание

Москва
Издательство МЦНМО
2021

УДК 510.5
ББК 22.12
Р17

Разборов А. А.
Алгебраическая сложность
Электронное издание
М.: МЦНМО, 2016
31 с.
ISBN 978-5-4439-2177-8

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. — 2-е изд., испр. — М.: МЦНМО, 2019. — ISBN 978-5-4439-2826-5.

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

ISBN 978-5-4439-2177-8

Теория алгебраической сложности была создана во многом благодаря усилиям немецкого математика Фолькера Штрассена [17], 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 [13]. 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 [16]), 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]:

$$\det\begin{pmatrix} x_{11} & x_{12} & \ldots & x_{1n} \\ \multicolumn{4}{c}{\dotfill} \\ x_{n1} & x_{n2} & \ldots & x_{nn}\end{pmatrix}=\sum_{\sigma\in S_n}\operatorname{sgn}(\sigma)x_{1\sigma(1)}\ldots x_{n\sigma(n)}.$$

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 B.

ствует он следующим образом. Пусть изначально нам дана матрица

$$\begin{pmatrix} x_{11} & x_{12} & \dots & x_{1n} \\ x_{21} & x_{22} & \dots & x_{2n} \\ \hdotsfor{4} \\ x_{n1} & x_{n2} & \dots & x_{nn} \end{pmatrix}.$$

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:

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

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:

$$\begin{pmatrix} x_{11} & x_{12} & \dots & x_{1n} \\ 0 & x'_{22} & \dots & x'_{2n} \\ \hdotsfor{4} \\ 0 & x'_{n2} & \dots & x'_{nn} \end{pmatrix}.$$

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

$$\begin{pmatrix} a_{11} & a_{12} & \dots & a_{1(n-1)} & a_{1n} \\ 0 & a_{22} & \dots & a_{2(n-1)} & a_{2n} \\ \hdotsfor{5} \\ 0 & 0 & \dots & 0 & a_{nn} \end{pmatrix}.$$

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

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

Но как этот определитель связан с определителем исходной матрицы? Мы делали только две операции: меняли местами строки в матрице и вычитали из одной строки другую. Предложение B.2 говорит о том, что первая операция меняет знак у определителя, а вторая — не меняет определитель вообще. Таким образом, мы заключаем:

$$\det \begin{pmatrix} x_{11} & x_{12} & \dots & x_{1n} \\ \multicolumn{4}{c}{\dotfill} \\ x_{n1} & x_{n2} & \dots & x_{nn} \end{pmatrix} = (-1)^t a_{11} a_{22} \dots a_{nn},$$

где t равно числу перемен местами строк в ходе выполнения алгоритма.

В методе Гаусса, как легко видеть, производится $O(n^3)$ умножений и $O(n^2)$ делений, что гораздо меньше $n!$. Из этого, благодаря результату Штрассена, вытекает существование алгоритма для вычисления определителя, который также использует $O(n^3)$ умножений, но зато вообще не использует деление. Оказывается, что число умножений можно сократить еще больше; об этом мы поговорим ниже в главе 4.

Вернемся к полиномам от одной переменной. На вход подаются коэффициенты $a_0, a_1, \dots, a_n \in \mathbb{R}$, и нужно вычислить полином:

$$a_0 + a_1 x + \dots + a_n x^n.$$

Рассмотрим следующий тривиальный способ вычисления.

1. Посчитаем все нужные степени x:

$$x, x^2, x^3, \dots, x^n.$$

На это у нас уйдет примерно n умножений.

2. Умножим степени x на соответствующие коэффициенты:

$$a_1 x, a_2 x^2, a_3 x^3, \dots, a_n x^n.$$

На это опять уйдет примерно n умножений.

3. Сложим все:

$$a_0 + a_1 x + \dots + a_n x^n.$$

Всего нам потребовалось примерно $2n$ умножений. Можно сэкономить n умножений, если воспользоваться схемой Горнера [10]. Представим наш полином в следующем виде:

$$a_0 + a_1 x + \dots + a_n x^n = a_0 + x(a_1 + x(\dots + x(a_{n-1} + x a_n)\dots)).$$

Легко удостовериться, что если вычислять выражение в правой части равенства, начиная со скобки $(a_{n-1} + x a_n)$, то нам потребуется всего $n+1$ умножение.

Мы привели несколько алгоритмов вычисления различных полиномов. Дадим общее определение, для которого все эти алгоритмы являются частными случаями. Итак, пусть задано, вообще говоря, k переменных $x_1, \ldots, x_k$ и полином от них $f(x_1, \ldots, x_k)$.

Определение 1.1. *Неветвящейся программой* называется последовательность строчек:

$$
\begin{aligned}
&f_1 = L_1(x_1, \ldots, x_k) R_1(x_1, \ldots, x_k),\\
&\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\\
&f_i = L_i(x_1, \ldots, x_k, f_1, \ldots, f_{i-1}) R_i(x_1, \ldots, x_k, f_1, \ldots, f_{i-1}),\\
&\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\\
&f_m = L_m(x_1, \ldots, x_k, f_1, \ldots, f_{m-1}) R_m(x_1, \ldots, x_k, f_1, \ldots, f_{m-1}),
\end{aligned}
\tag{1}
$$

где для всякого i от 1 до m выражения L_i и R_i являются линейными функциями от $x_1, \ldots, x_k, f_1, \ldots, f_{i-1}$, то есть выражениями вида

$$c_0 + c_1 x_1 + \ldots + c_k x_k + d_1 f_1 + \ldots + d_{i-1} f_{i-1},$$

где, в свою очередь, $c_0, c_1, \ldots, c_k, d_1, \ldots, d_{i-1}$ — некоторые константы.

Длиной неветвящейся программы называется количество строчек в ней, для программы (1) это количество равно m. Неветвящаяся программа вычисляет полином f, если $f_m = f$.

Определение 1.2. *Алгебраической сложностью* полинома f, которую обозначают через $L(f)$, называется длина кратчайшей неветвящейся программы, вычисляющей f.

Вооруженные этим определением, мы можем задаться вопросом — а существуют ли вообще сложные полиномы? Давайте пока считать, что f — это полином степени n от одной переменной, т. е.

$$f(x) = a_0 + a_1 x + \ldots + a_n x^n.$$

Какой длины m нам достаточно, чтобы неветвящиеся программы длины m порождали всевозможные такие полиномы? Спросим себя, от скольких параметров зависит неветвящаяся программа длины m. Первая строчка программы зависит от четырех параметров, поскольку L_1, R_1 — линейные функции от x, а для того чтобы каждую из них задать, требуется 2 числа. Легко теперь понять, что вторая строчка программы зависит от 6 параметров, третья — от 8 параметров и т. д. Если просуммировать по всем строчкам программы, получится, что программа длины m зависит не более чем от

$$t = 4 + 6 + 8 + \ldots + (2m + 2) = m^2 + 3m$$

параметров. Обозначим эти параметры через $p_1, \ldots, p_t$. Посмотрим, что из себя представляет f_m. Понятно, что f_m — это полином от x, причем при любой степени x в этом полиноме стоит коэффициент, в свою очередь являющийся полиномом от параметров программы, т. е. от $p_1, \ldots, p_t$. Иными словами, имеется представление

$$f_m = b_0(p_1, \ldots, p_t) + b_1(p_1, \ldots, p_t)x + \ldots + b_n(p_1, \ldots, p_t)x^n,$$

где $b_0, b_1, \ldots, b_n$ — полиномы. Поскольку требуется, чтобы f_m равнялось f, мы получаем систему уравнений:

$$\begin{cases} a_0 = b_0(p_1, \ldots, p_t), \\ a_1 = b_1(p_1, \ldots, p_t), \\ \ldots\ldots\ldots\ldots\ldots\ldots \\ a_n = b_n(p_1, \ldots, p_t), \end{cases}$$

причем требуется, чтобы она была разрешима для любого набора $a_0, \ldots$ $\ldots, a_n \in \mathbb{R}$. В этой системе $n+1$ уравнение и t неизвестных. Число уравнений не должно превосходить число неизвестных (это, конечно, требует доказательства), т. е. должно выполняться неравенство $n+1 \leqslant t = m^2 + 3m$. Это означает, что невозможно неветвящимися программами длины m вычислить все полиномы степени m^2+3m, из чего вытекает следующая теорема:

Теорема 1.1. *Существует полином степени n от одной переменной, алгебраическая сложность которого не меньше $\Omega(\sqrt{n})$ (необходимые сведения об обозначении $\Omega(\cdot)$ можно найти в приложении D).*

Патерсон и Стокмейер [11] доказали, что эта оценка точна, т. е. любой полином степени n можно вычислить неветвящейся программой длины $\sqrt{O(n)}$.

Мы доказали существование сложного полинома. Можно ли предъявить какой-нибудь сложный полином в явном виде, т. е. доказать нижнюю оценку сложности для какого-нибудь конкретного полинома? Эта задача остается открытой уже больше 30 лет, и почти никакого прогресса в ее решении не имеется.

Выдвинуто несколько гипотез относительно сложности некоторых полиномов. Рассмотрим разложение функции e^x в ряд:

$$e^x = 1 + x + \frac{x^2}{2} + \ldots + \frac{x^n}{n!} + \ldots$$

Неизвестны ни нижние оценки сложности его частичных сумм, ни верхние, кроме тривиальных. Выдвинуто предположение, что для вычисления суммы первых n слагаемых этого ряда требуется $\Omega(n)$ умножений. Та же самая ситуация с частичными суммами ряда для функции $\ln(1-x)$:

$$\ln(1-x) = -x - \frac{x^2}{2} - \ldots - \frac{x^n}{n} - \ldots$$

2. Полиномы от многих переменных

Про полиномы от многих переменных известно больше. Рассмотрим, например, следующий полином:

$$x_1^2+x_2^2+\ldots+x_k^2.$$

Если нам разрешено пользоваться комплексными числами, то нетрудно догадаться, как его вычислить за $\frac{k}{2}$ умножений:

$$x_1^2+x_2^2+\ldots+x_k^2=(x_1+ix_2)(x_1-ix_2)+(x_3+ix_4)(x_4-ix_4)+\ldots$$

Однако если мы остаемся над полем вещественных чисел, то алгебраическая сложность этого полинома не меньше k, и мы это сейчас докажем.

Теорема 2.1.

$$L(x_1^2+x_2^2+\ldots+x_k^2)\geqslant k.$$

Доказательство. Предположим, что кратчайшая неветвящаяся программа, вычисляющая $x_1^2+x_2^2+\ldots+x_k^2$, имеет длину $m<k$. Пусть эта программа имеет вид (1). Воспользуемся так называемым *методом подстановок*, разработанным Паном в [21] для доказательства оптимальности метода Горнера и позже развитым в [7, 8, 14, 19]. Рассмотрим первую строчку программы:

$$f_1=L_1(x_1,\ldots,x_k)R_1(x_1,\ldots,x_k).$$

Если обе функции L_1, R_1 — константы, то f_1 — тоже константа. Тогда первую строчку можно вычеркнуть, а во все оставшиеся строчки программы подставить значение f_1. Пусть без ограничения общности функция L_1 не является константой и зависит от x_{i_1} (т. е. в выражение для L_1 переменная x_{i_1} входит с ненулевым коэффициентом). Будем для простоты считать, что $i_1=1$. Тогда из уравнения

$$L_1(x_1,\ldots,x_k)=0$$

можно выразить x_1 через остальные переменные:

$$x_1=T_1(x_2,\ldots,x_k),$$

где T_1 — какая-то линейная функция.

Подставим во все следующие строчки программы вместо x_1 функцию $T_1(x_2,\ldots,x_k)$, вместо f_1 — ноль, а первую строчку просто сотрем. Мы получим неветвящуюся программу длины $m-1$ для вычисления полинома

$$(T_1(x_2,\ldots,x_k))^2+x_2^2+\ldots+x_k^2.$$

Повторим ту же самую процедуру для первой строчки новой программы и так далее. Мы будем последовательно выражать переменные $x_1, x_2, \ldots$ через оставшиеся так, чтобы занулялась очередная строчка программы (без ограничения общности можно считать, что именно переменная x_i выражается через остальные переменные на i-м шаге, поскольку порядок переменных не важен). Рассмотрим последнюю строчку программы. Поскольку по предположению $k > m$, то эту строчку все еще можно занулить тем же способом. Однако эта строчка представляет собой программу длины 1, вычисляющую многочлен вида

$$(T_1(x_{m+1}, \ldots, x_k))^2 + \ldots + (T_m(x_{m+1}, \ldots, x_k))^2 + x_{m+1}^2 + \ldots + x_k^2,$$

где $T_1, \ldots, T_{m-1}$ — некоторые линейные функции. Получается, что этот многочлен зануляется на любом наборе $x_{m+1}, \ldots, x_k$, что невозможно для суммы квадратов (если мы подставим $x_k \neq 0$, то значение будет строго положительно вне зависимости от других переменных). $\square$

Ниже речь пойдет о более сильном методе, при помощи которого можно получать нижние оценки алгебраической сложности многочленов. Этот метод был разработан в 70—80-х годах прошлого века, и к настоящему моменту неизвестно, можно ли его улучшить.

Прежде всего, давайте немного расширим нашу модель. До этого речь шла о вычислении одного полинома f и его алгебраической сложности $L(f)$. Рассмотрим более сложную ситуацию, когда у нас есть несколько полиномов $p_1, \ldots, p_m$ и мы хотим вычислить их все при помощи одной неветвящейся программы. Это просто означает, что для каждого полинома в программе (1) найдется строчка (не обязательно последняя), равная этому полиному. Длину кратчайшей такой программы, как и раньше, обозначим через $L(p_1, \ldots, p_m)$.

Пусть $p_1 = x_1^d, \ldots, p_m = x_m^d$. Это попросту означает, что нам дано m чисел и мы хотим возвести их в степень d. Первое, что приходит в голову, — это возвести каждое число в степень d по отдельности. Поскольку возводить в степень d мы уже умеем за $O(\log_2(d))$ умножений, у нас получается следующая оценка:

$$L(x_1^d, \ldots, x_m^d) = O(m \log_2(d)).$$

Здравый смысл подсказывает, что ничего лучше, чем вычислить каждый многочлен в отдельности оптимальным для него способом, придумать нельзя. Однако бывают случаи, когда подобная интуиция обманчива (о них речь пойдет ниже). А доказать, что не существует более быстрого способа, чем последовательное вычисление, вообще говоря, очень непросто. Для этого нам будет полезно ввести понятие *геометрической степени*. Мы не будем давать строгое определение геометрической степени для системы полиномов в общем случае. Мы сделаем это для частного случая, когда количество полиномов совпадает с количеством переменных, как это имеет место с системой $\{x_1^d, \ldots, x_m^d\}$. Итак, пусть у нас имеется m полиномов от m переменных:

$$p_1(x_1, \ldots, x_m), \ldots, p_m(x_1, \ldots, x_m),$$

которые для простоты будут предполагаться алгебраически независимыми. Геометрической степенью системы полиномов $\{p_1, \ldots, p_m\}$ называется максимальное возможное количество решений (над полем комплексных чисел) системы:

$$\begin{cases} \xi_1 = p_1(x_1, \ldots, x_m), \\ \xi_2 = p_2(x_1, \ldots, x_m), \\ \ldots\ldots\ldots\ldots\ldots\ldots \\ \xi_m = p_m(x_1, \ldots, x_m), \end{cases}$$

где максимум берется по всем возможным значениям $\xi_1, \ldots, \xi_m \in \mathbb{C}$. Геометрическая степень обозначается через $W(p_1, \ldots, p_m)$; при $m = 1$ она совпадает с обычной степенью (ср. с теоремой А.1). Имеет место следующая теорема:

Теорема 2.2 (Штрассен [15]). *Для всякой системы полиномов вида*

$$p_1(x_1, \ldots, x_m), \ldots, p_m(x_1, \ldots, x_m)$$

выполнено неравенство

$$L(p_1, \ldots, p_m) \geqslant \log_2 \left(W(p_1, \ldots, p_m)\right).$$

Применим этот результат к системе полиномов $\{x_1^d, \ldots, x_m^d\}$. Рассмотрим систему уравнений:

$$\begin{cases} 1 = x_1^d, \\ 1 = x_2^d, \\ \ldots\ldots\ldots \\ 1 = x_m^d. \end{cases}$$

Нетрудно понять, что над полем комплексных чисел у этой системы ровно d^m (каждый из $x_1, \ldots, x_m$ обязан быть комплексным корнем степени d из единицы, а таких корней ровно d). Из этого вытекает, что

$$L(x_1^d, \ldots, x_m^d) \geqslant \log_2(W(p_1, \ldots, p_m)) \geqslant \log_2(d^m) = m\log_2(d).$$

Как мы видим, нижняя оценка $L(x_1^d, \ldots, x_m^d)$ по порядку совпала с верхней.

Мы доказали (сославшись на теорему 2.2), что нельзя совместно вычислить систему полиномов $x_1^d, \ldots, x_m^d$ быстрее, чем за $m\log_2(d)$ умножений. Что будет, если вместо значений x_i^d для всех i нам нужно вычислить только их сумму? Ясно, что эта задача не сложнее, поскольку если мы уже вычислили $x_1^d, \ldots, x_m^d$, то сложить их уже не составляет большого труда. Кроме того, в первой части мы получили оценку

$$L(x_1^2 + \ldots + x_m^2) \geqslant m,$$

что совпадает с известной нам нижней оценкой на $L(x_1^d, \ldots, x_m^d)$ при $d = 2$. Оказывается, аналогичная оценка выполнена и при больших d:

$$L(x_1^d + \ldots + x_m^d) \geqslant \Omega(m\log_2(d)).$$

Несмотря на то, что эти результаты кажутся очень похожими, их разделяет порядка 10 лет.

Здесь нам придется сослаться без доказательства на один из красивейших фактов в этой области.

Теорема 2.3 (Баур, Штрассен [1]). *Пусть $f(x_1, \ldots, x_m)$ — произвольный полином. Тогда*

$$L\left(f, \frac{\partial f}{\partial x_1}, \ldots, \frac{\partial f}{\partial x_m}\right) \leqslant 3L(f).$$

Грубо говоря, эта теорема утверждает, что если при помощи некоторой программы можно вычислить полином, то при помощи лишь в три раза более длинной программы можно вычислить одновременно полином и все его частные производные.

Пользуясь этой теоремой, уже очень легко получить нижнюю оценку на $L(x_1^d + \ldots + x_m^d)$:

$$\begin{aligned} L(x_1^d + \ldots + x_m^d) &\geqslant \frac{1}{3} L(x_1^d + \ldots + x_m^d, dx_1^{d-1}, \ldots, dx_m^{d-1}) \geqslant \\ &\geqslant \frac{1}{3} L(dx_1^{d-1}, \ldots, dx_m^{d-1}) = \\ &= \frac{1}{3} L(x_1^{d-1}, \ldots, x_m^{d-1}) \geqslant \frac{1}{3} m\log_2(d-1) = \Omega(m\log_2(d)). \end{aligned}$$

3. Перемножение полиномов и вычисление билинейных форм

До сих пор мы занимались вычислением полиномов. Займемся немного другой вещью — будем полиномы перемножать. Пусть имеется два полинома:

$$p(x) = a_0 + a_1 x + \ldots + a_n x^n, \quad q(x) = b_0 + b_1 x + \ldots + b_n x^n,$$

и мы хотим вычислить коэффициенты полинома $p(x)q(x)$, т. е.

$$(a_0+a_1x+\ldots+a_nx^n)(b_0+b_1x+\ldots+b_nx^n)=$$
$$=a_0b_0+(a_1b_0+a_0b_1)x+\ldots+a_nb_nx^{2n}.$$

Здесь несущественно, что степени полиномов равны. Действительно, если у одного из многочленов степень меньше, чем у другого, то мы можем положить в нем равными нулю коэффициенты при степенях x, превосходящих его степень.

Ясно, что коэффициент c_i при x^i в произведении $p(x)q(x)$ задается формулой

$$c_i=\sum_{t=\max\{0,i-n\}}^{\min\{n,i\}} a_tb_{i-t}.$$

Как видно, он является полиномом от $a_0, \ldots, a_n, b_0, \ldots, b_n$. Формально говоря, нас будет интересовать алгебраическая сложность системы полиномов $c_0, c_1, \ldots, c_{2n}$. Эта система называется *системой билинейных форм*.

Тривиальный алгоритм предписывает найти произведение всех пар a_ib_j. На это уходит $(n+1)^2$ умножений, из чего мы заключаем, что

$$L(c_0, c_1, \ldots, c_{2n})=O(n^2).$$

На самом деле можно обойтись всего $O(n)$ умножениями. Для этого нужно вспомнить, что полином r степени m можно однозначно задать значениями в $m+1$ различных точках:

$$(x_0, r(x_0)),\ (x_1, r(x_1)),\ \ldots,\ (x_m, r(x_m)).$$

Действительно, если бы два разных полинома степени m совпадали в $m+1$ различных точках, то тогда их разность имела бы $m+1$ различных корней, что может быть, только когда она равна нулю (теорема Безу А.1).

Теперь зафиксируем произвольные $2n+1$ точек, например:

$$0, 1, 2, \ldots, 2n.$$

Мы хотим вычислить коэффициенты полинома $r(x)=p(x)q(x)$. Сначала вычислим

$$p(0), p(1), \ldots, p(2n), q(0), q(1), \ldots, q(2n).$$

На это у нас не уйдет *ни одного* умножения, поскольку значение полинома в фиксированной точке есть линейная функция от его коэффициентов. Далее производится $2n+1$ умножение: находятся значения $r(x)$

в точках $0, 1, \ldots, 2n+1$ по очевидной формуле

$$r(i) = p(i)q(i).$$

Теперь по значениям $r(x)$ в $2n+1$ различных точках нам нужно восстановить его коэффициенты. Воспользуемся *интерполяционной формулой Лагранжа*:

$$r(x) = \sum_{i=0}^{2n} r(i) \frac{\prod\limits_{j=0, j\neq i}^{2n} (x-j)}{\prod\limits_{j=0, j\neq i}^{2n} (i-j)}.$$

В справедливости этой формулы легко убедиться: справа стоит многочлен степени не выше $2n$, совпадающий с r в точках $0, 1, \ldots, 2n$ (последнее проверяется просто подстановкой). Итак, мы представили $r(x)$ в виде *линейного* по $r(0), r(1), \ldots, r(2n)$ (которые у нас уже вычислены) выражения. Чтобы теперь найти его коэффициенты, не требуется ни одного умножения (напоминаем, что умножение на константы бесплатно, см. с. 3). Мы заключаем:

$$L(c_0, c_1, \ldots, c_{2n}) = O(n).$$

До сих пор сложение мы считали бесплатной операцией. Однако от выбора системы точек, по значениям в которых восстанавливаются коэффициенты полинома, сильно зависит, сколько нам потребуется сложений в описанном только что алгоритме. Чтобы указать наилучшую с этой точки зрения систему точек, поговорим теперь немного об *аддитивной сложности*.

Пусть имеется фиксированная матрица

$$A = \begin{pmatrix} a_{11} & a_{12} & \ldots & a_{1n} \\ \multicolumn{4}{c}{\dotfill} \\ a_{n1} & a_{n2} & \ldots & a_{nn} \end{pmatrix}$$

и мы хотим вычислить ее произведение на вектор переменных $x = (x_1, \ldots, x_n)$. Иными словами, наша задача — вычислить следующие n выражений:

$$\begin{gathered} a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n, \\ a_{21}x_1 + a_{22}x_2 + \ldots + a_{2n}x_n, \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ a_{n1}x_1 + a_{n2}x_2 + \ldots + a_{nn}x_n. \end{gathered}$$

Делается это (по аналогии с неветвящимися программами) при помощи следующей последовательности строчек-инструкций:

$$
\begin{aligned}
&f_1 = \alpha_1 u_1 + \beta_1 v_1,\\
&\dots\dots\dots\dots\dots\dots\\
&f_i = \alpha_i u_i + \beta_i v_i,\\
&\dots\dots\dots\dots\dots\dots\\
&f_m = \alpha_m u_m + \beta_m v_m.
\end{aligned}
\tag{2}
$$

Здесь для всякого i числа $\alpha_i, \beta_i \in \mathbb{R}$ — какие-то константы, а u_i, v_i — либо переменные, либо ранее вычисленные $f_1, \dots, f_{i-1}$, т. е.

$$u_i, v_i \in \{x_1, \dots, x_n, f_1, \dots, f_{i-1}\}.$$

Для всякого $j \in \{1, 2, \dots, n\}$ должна найтись строчка в программе под номером i такая, что

$$f_i = a_{j1}x_1 + a_{j2}x_2 + \dots + a_{jn}x_n.$$

Как обычно, нас интересует самая короткая последовательность инструкций. Длина самой короткой последовательности, вычисляющая данную матрицу, называется *аддитивной сложностью* данной матрицы.

Мы займемся одной из самых важных матриц. Она будет связана с декодированием полиномов степени $n-1$ по их значениям в n точках, выбранных специальным образом. Зафиксируем w — произвольный *первообразный* корень n-й степени из единицы (который существует согласно предложению C.2) и для каждого $k \in \{0, 1, \dots, n-1\}$ положим

$$a_k = w^k.$$

Каждое a_k, конечно, тоже будет корнем степени n из единицы:

$$a_k^n = w^{kn} = 1^k = 1.$$

Пусть наша матрица A имеет вид

$$A = \begin{pmatrix} 1 & a_0 & a_0^2 & \dots & a_0^{n-1} \\ 1 & a_1 & a_1^2 & \dots & a_1^{n-1} \\ \dots & \dots & \dots & \dots & \dots \\ 1 & a_{n-1} & a_{n-1}^2 & \dots & a_{n-1}^{n-1} \end{pmatrix}.$$

Иными словами, на пересечении k-й строки и j-го столбца (если нумеровать их с нуля) стоит $a_k^j = w^{kj}$. Эта матрица называется *матрицей дискретного преобразования Фурье*.

Что будет, если умножить эту матрицу на вектор $(c_0, c_1, \ldots, c_{n-1})$? Легко проверить, что на месте k-й координаты вектора

$$A\begin{pmatrix} c_0 \\ c_1 \\ \ldots \\ c_{n-1} \end{pmatrix}$$

будет стоять значение полинома

$$r(x) = c_0 + c_1 x + \ldots + c_{n-1}x^{n-1}$$

в точке a_k. Чтобы по значениям $r(x)$ в точках $a_0, \ldots, a_{n-1}$ найти коэффициенты полинома, нам требуется решить линейную систему на $c_0, \ldots, c_{n-1}$ следующего вида:

$$\begin{pmatrix} r(a_0) \\ r(a_1) \\ \ldots \\ r(a_{n-1}) \end{pmatrix} = A\begin{pmatrix} c_0 \\ c_1 \\ \ldots \\ c_{n-1} \end{pmatrix}. \tag{3}$$

Чтобы ее решить, вообще говоря, нужно обратить матрицу A. Оказывается, что матрица дискретного преобразования Фурье обращается очень просто — обратная к ней матрица совпадает с ней с точностью до комплексного сопряжения и до множителя. Точнее говоря,

$$A^{-1} = \frac{1}{n}A^*.$$

Давайте проверим это. На пересечении k-й строки и j-го столбца в матрице $A\frac{A^*}{n}$ по определению умножения матриц будет стоять число

$$\frac{1}{n}\sum_{s=0}^{n-1} w^{ks}w^{-sj} = \frac{1}{n}\sum_{s=0}^{n-1}(w^{k-j})^s.$$

Если $k = j$, то очевидно, что указанная сумма равна 1. Если $k \neq j$, т. е. $w^{k-j} \neq 1$ (здесь мы пользуемся тем, что w — именно *первообразный* корень), то правомерно применить формулу суммирования геометрической прогрессии:

$$\frac{1}{n}\sum_{s=0}^{n-1}(w^{k-j})^s = \frac{1-w^{(k-j)n}}{n(1-w^{k-j})} = 0.$$

Итак, мы показали, что произведение $A\frac{A^*}{n}$ равно единичной матрице, что и означает, что $A^{-1} = \frac{1}{n}A^*$. Теперь систему (3) мы можем переписать в следующем эквивалентном виде:

$$\begin{pmatrix} c_0 \\ c_1 \\ \dots \\ c_{n-1} \end{pmatrix} = \frac{A^*}{n} \begin{pmatrix} r(x_0) \\ r(x_1) \\ \dots \\ r(x_{n-1}) \end{pmatrix}. \tag{4}$$

Получилось, что система точек $a_0, \dots, a_{n-1}$ подобрана настолько хорошо, что теперь, чтобы найти коэффициенты полинома, нам не требуется обращать матрицу A, а требуется только умножить вектор значений полинома в точках $x_0, \dots, x_{n-1}$ на матрицу $\frac{A^*}{n}$.

Насколько быстро (с точки зрения числа сложений) это можно сделать? Иными словами, чему равна аддитивная сложность матрицы быстрого преобразования Фурье A (аддитивная сложность у матрицы $\frac{A^*}{n}$, как нетрудно понять, такая же)? Тривиальный алгоритм умножения матрицы на вектор по определению этого произведения дает оценку $O(n^2)$, он годится для любой матрицы. Однако для матрицы дискретного преобразования Фурье существует алгоритм, имеющий сложность $O(n\log_2(n))$ — он называется *быстрым дискретным преобразованием Фурье* [3, 6, 12, 20]. Несмотря на то, что этот алгоритм добивается существенного ускорения по сравнению с тривиальным алгоритмом, до сих пор неизвестно, существует ли алгоритм сложности меньше, чем $O(n\log_2(n))$.

Кое-какие нижние оценки доказать все-таки удается. Для примера отметим один факт, показанный Моргенштерном в работе [9]. Быстрое преобразование Фурье обладает следующим хорошим свойством: коэффициенты α_i, β_i из последовательности инструкций вида (2) для быстрого преобразования Фурье не превышают по модулю двойки. Оказывается, что любой алгоритм, у которого эти коэффициенты не превышают некоторой константы, имеет сложность не меньше $\Omega(n\log_2(n))$, так что если алгоритм меньшей сложности и существует, то он должен быть устроен существенно по-другому.

4. Умножение матриц

Теперь мы займемся еще одной, важнейшей с прикладной точки зрения, задачей — задачей умножения матриц:

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

По определению, в произведении на пересечении i-й строки и k-го столбца стоит выражение: $\sum_{j=1}^{n} x_{ij}y_{jk}$. Формально говоря, нас интересует задача совместного вычисления системы полиномов

$$\left\{\sum_{j=1}^{n} x_{ij}y_{jk} \,|\, i,k \in \{1,\ldots,n\}\right\},$$

и алгебраическая сложность этой системы

$$L\left(\left\{\sum_{j=1}^{n} x_{ij}y_{jk} \,|\, i,k \in \{1,\ldots,n\}\right\}\right).$$

Как обычно, начнем с рассмотрения тривиального алгоритма. Тривиальный алгоритм вычисляет произведения вида $x_{ij}y_{jk}$ по всем $i, j, k \in$ $\in \{1,\ldots,n\}$. Таких произведений, очевидно, n^3. Таким образом:

$$L\left(\left\{\sum_{j=1}^{n} x_{ij}y_{jk} \,|\, i,k \in \{1,\ldots,n\}\right\}\right) = O(n^3).$$

Можно умножать матрицы и быстрее. Первым это выяснил Штрассен. Его алгоритм опирается на следующий факт: оказывается, произведение матриц 2×2

$$\begin{pmatrix} x_{11} & x_{12} \\ x_{21} & x_{22} \end{pmatrix} \times \begin{pmatrix} y_{11} & y_{12} \\ y_{21} & y_{22} \end{pmatrix}$$

можно вычислить, сделав всего 7 умножений, а не 8. Мы не будем выписывать явные формулы, но посмотрим на философское значение этого факта. Можно доказать, что нельзя умножить матрицу $\begin{pmatrix} x_{11} & x_{12} \\ x_{21} & x_{22} \end{pmatrix}$ на столбец $\begin{pmatrix} y_{11} \\ y_{21} \end{pmatrix}$, используя меньше 4 умножений. Конечно, аналогичное утверждение верно и для столбца $\begin{pmatrix} y_{12} \\ y_{22} \end{pmatrix}$. Однако умножить оба эти столбца мы можем за 7 умножений, что меньше $2 \cdot 4$. Этот пример показывает, что сложность выполнения n независимых однотипных задач, деленная на n, может быть меньше сложности одной задачи.

Для матриц 2×2 мы сэкономили 1 умножение. Для матриц большего размера можно сэкономить больше умножений. Рассмотрим сначала матрицы 4×4. Можно разбить обе матрицы на 4 блока размера 2×2:

$$\begin{pmatrix} X_{11} & X_{12} \\ X_{21} & X_{22} \end{pmatrix}, \quad \begin{pmatrix} Y_{11} & Y_{12} \\ Y_{21} & Y_{22} \end{pmatrix}.$$

Далее мы применим метод Штрассена для умножения матриц 2×2, в которых вместо чисел стоят матрицы 2×2. Нам понадобится 7 умножений матриц 2×2, причем каждое такое специальное умножение мы проделываем при помощи того же метода Штрассена, используя 7 уже обычных умножений. Всего получается $7^2 = 49$ умножений.

Рассмотрим теперь матрицы произвольного размера $n \times n$, правда, для простоты мы будем считать, что n — степень двойки, т. е. $n = 2^k$. Понятно, что тем же самым способом, при помощи которого мы научились умножать матрицы 4×4 за 49 умножений, можно умножать матрицы размера $2^k \times 2^k$ за 7^k умножений. Выразим это число умножений через n:

$$7^k = 7^{\log_2(n)} = 2^{\log_2(7)\log_2(n)} = n^{\log_2(7)}.$$

Понятно, что $\log_2(7) < 3$, т. е. мы получили способ умножения произвольных матриц сложности намного меньше $O(n^3)$. Мы заключаем, что

$$L\left(\left\{\sum_{j=1}^{n} x_{ij}y_{jk} \,|\, i, k \in \{1, \ldots, n\}\right\}\right) = O(n^{\log_2(7)}) \approx O(n^{2,807}).$$

Но и это не предел. Были построены еще более эффективные алгоритмы, кроме того, была создана красивейшая теория вокруг этой задачи.

Через w обозначается так называемый *показатель матричного умножения*, т. е. минимальное действительное число ω, для которого

$$L\left(\left\{\sum_{j=1}^{n} x_{ij}y_{jk} \,|\, i, k \in \{1, \ldots, n\}\right\}\right) = n^{\omega + o(1)}.$$

Результат Штрассена показывает, что $\omega \leqslant \log_2(7) \approx 2{,}807$. Не очень трудно показать, что $\omega \geqslant 2$. На настоящий момент известно, что $\omega \leqslant$ $\leqslant 2{,}38$. Эта оценка принадлежит Д. Копперсмиту и Ш. Винограду. Недавно было получено новое доказательство этого результата, удивительным образом связывающее умножение матриц со свойствами групп перестановок. В данной брошюре мы можем сказать лишь пару общих слов об этом доказательстве. Существование алгоритма в нем вытекает из существования конечных групп перестановок с некоторыми экстремальными свойствами. Кроме того, оказывается (см. [4]), что если бы существовали группы перестановок с еще более экстремальными свойствами (что пока еще не удается доказать), то существовал бы алгоритм для перемножения матриц сложности $O(n^2)$.

С тех пор, как была прочитана данная лекция, ситуация немного изменилась, и появились новые результаты. В работе [18] показана оценка $\omega \leqslant 2{,}3727$. Также свежей разработкой является [5], где хотя и не показано новых оценок, но намечен возможный путь к улучшению предыдущих результатов.

В недавней статье [2] предпринята вполне разумная попытка классификации существующих подходов к построению быстрых алгоритмов для матричного умножения. Доказанные в этой статье результаты делают более правдоподобной гипотезу о том, что на самом деле имеет место $\omega > 2$.

5. Перманент и VNP-полнота

До сих пор мы решали важные, но не очень связанные друг с другом задачи из теории алгебраической сложности. В последнем разделе мы поговорим о том, что связывает алгебраическую сложность с другими разделами теории сложности.

Вспомним формулу определителя:

$$\det\begin{pmatrix} x_{11} & x_{12} & \dots & x_{1n} \\ x_{21} & x_{22} & \dots & x_{2n} \\ \hdotsfor{4} \\ x_{n1} & x_{n2} & \dots & x_{nn} \end{pmatrix} = \sum_{\sigma\in S_n} (-1)^\sigma x_{1\sigma(1)}x_{2\sigma(2)}\dots x_{n\sigma(n)}.$$

Мы уже выяснили, что сложность вычисления определителя матрицы $n\times n$ есть $O(n^3)$. Давайте «упростим» себе задачу, беря произведения вида $x_{1\sigma(1)}x_{2\sigma(2)}\dots x_{n\sigma(n)}$ всегда с плюсом, вне зависимости от знака перестановки σ. Полученная функция называется *перманентом*:

$$\operatorname{perm}\begin{pmatrix} x_{11} & x_{12} & \dots & x_{1n} \\ x_{21} & x_{22} & \dots & x_{2n} \\ \hdotsfor{4} \\ x_{n1} & x_{n2} & \dots & x_{nn} \end{pmatrix} = \sum_{\sigma\in S_n} x_{1\sigma(1)}x_{2\sigma(2)}\dots x_{n\sigma(n)}.$$

Оказывается, перманент вычислить значительно сложнее, чем определитель. Гипотеза состоит в том, что не бывает неветвящихся программ для вычисления перманента матрицы размера $n\times n$ длины, ограниченной некоторым полиномом от n. Гипотеза эта не доказана, однако есть достаточно убедительные косвенные аргументы в ее пользу. Этим аргументом служит теория алгебраической NP-полноты[1], разработанная Лесли Вэлиантом.

[1] Краткие сведения о теории NP-полноты см., например, в гл. 8 книги *С. Дасгупта*, *Х. Пападимитриу*, *У. Вазирани*. Алгоритмы. М.: МЦНМО, 2014.

В качестве затравки мы укажем следующий факт:

Теорема 5.1. *Пусть f — некоторый полином сложности не выше t, т. е. $L(f) \leqslant t$. Тогда f можно представить в виде определителя некоторой матрицы A:*

$$f = \det A,$$

причем размер матрицы A ограничен некоторым полиномом от t, а элементы матрицы A есть либо переменные, от которых зависит f, либо коэффициенты.

Мы будем рассматривать не полиномы, а семейства полиномов. Семейство полиномов $\{f_n\}_{n\in\mathbb{N}}$ называется *полиномиально ограниченным*, если степень f_n и число переменных, от которых зависит f_n, ограничено некоторым полиномом от n.

Определение 5.1. Классом VP называется множество всех полиномиально ограниченных семейств полиномов $\{f_n\}$, для которых выполнено неравенство

$$L(f_n) \leqslant p(n),$$

где $p(n)$ — некоторый полином, зависящий от семейства.

Например, пусть $\det_n$ — полином, равный определителю матрицы размера $n \times n$, т. е.:

$$\det_n(x_{11}, \ldots, x_{nn}) = \det \begin{pmatrix} x_{11} & x_{12} & \ldots & x_{1n} \\ x_{21} & x_{22} & \ldots & x_{2n} \\ \ldots & \ldots & \ldots & \ldots \\ x_{n1} & x_{n2} & \ldots & x_{nn} \end{pmatrix}.$$

Тогда семейство $\{\det_n\}_{n\in\mathbb{N}}$ принадлежит классу VP. Действительно, степень $\det_n$ равна n, а количество переменных, от которых он зависит, равно n^2. Это означает, что семейство $\{\det_n\}_{n\in\mathbb{N}}$ является полиномиально ограниченным. Кроме того, мы умеем вычислять определитель за $O(n^3)$ умножений, т. е.

$$L(\det_n) \leqslant Cn^3,$$

где C — некоторая константа. По определению это означает, что

$$\{\det_n\}_{n\in\mathbb{N}} \in \mathrm{VP}.$$

VP — это алгебраический аналог класса P из теории сложности вычислений (буква V соответствует первой букве фамилии Лесли Вэлианта). Как определить алгебраический аналог класса NP? Самым естественным оказывается следующий способ.

Определение 5.2. Полиномиально ограниченное семейство полиномов $\{f_n(x_1, \ldots, x_{m(n)})\}$ принадлежит классу VNP, если для него найдется семейство полиномов $\{g_n(x_1, \ldots, x_{m(n)}, y_1, \ldots, y_{k(n)})\}$ из класса VP такое, что выполнено равенство

$$f_n(x_1, \ldots, x_{m(n)}) = \sum_{e \in \{0,1\}^{k(n)}} g_n(x_1, \ldots, x_{m(n)}, e_1, \ldots, e_{k(n)}).$$

Суммирование ведется по всем векторам e из нулей и единиц длины $k(n)$, а e_i равно значению i-й координаты вектора e.

Как и в случае с определителем, через perm_n обозначим полином, равный перманенту матрицы размера $n \times n$, т. е.

$$\mathrm{perm}_n(x_{11}, \ldots, x_{nn}) = \mathrm{perm} \begin{pmatrix} x_{11} & x_{12} & \ldots & x_{1n} \\ x_{21} & x_{22} & \ldots & x_{2n} \\ \ldots & \ldots & \ldots & \ldots \\ x_{n1} & x_{n2} & \ldots & x_{nn} \end{pmatrix}.$$

Совершенно очевидно, что семейство $\{\mathrm{perm}_n\}_{n\in\mathbb{N}}$ является полиномиально ограниченным. Кроме того, оказывается, что это семейство принадлежит классу VNP.

Предложение 5.1.

$$\{\mathrm{perm}_n\}_{n\in\mathbb{N}} \in \mathrm{VNP}.$$

Доказательство. Полином perm_n зависит от переменных $x_{11}, \ldots, x_{nn}$. Заведем еще n^2 переменных $y_{11}, \ldots, y_{nn}$. По определению положим

$$p_{ij} = y_{i1} \ldots y_{i(j-1)}(1 - y_{ij}) y_{i(j+1)} \ldots y_{in}.$$

Например,

$$p_{11} = (1 - y_{11}) y_{12} \ldots y_{1n}.$$

Все p_{ij} являются полиномами степени n от переменных $y_{11}, \ldots, y_{nn}$. Положим

$$g_n(x_{11}, \ldots, x_{nn}, y_{11}, \ldots, y_{nn}) = \prod_{i=1}^{n} (p_{1i} x_{i1} + \ldots + p_{ni} x_{in}).$$

Покажем, что семейство $\{g_n\}_{n\in\mathbb{N}}$ лежит в классе VP. Для этого надо научиться вычислять полином g_n за полиномиальное от n число умножений. Чтобы вычислить все p_{ij}, нам потребуется порядка n^3 умножений — на каждый p_{ij} уходит n умножений, а всего их n^2. Далее надо вычислить все скобки вида

$$(p_{1i} x_{i1} + \ldots + p_{ni} x_{in}),$$

на что мы потратим n^2 умножений. Наконец, надо перемножить все скобки, на что тратится еще n умножений.

Нам остается доказать следующее равенство:

$$\mathrm{perm}_n(x_{11}, \ldots, x_{nn}) = \sum_{e\in\{0,1\}^{n^2}} g_n(x_{11}, \ldots, x_{nn}, e_{11}, \ldots, e_{nn}).$$

Раскроем скобки в правой части по определению полинома g_n. Полученное выражение разобьется на слагаемые вида

$$\sum_{e\in\{0,1\}^{n^2}} p_{k_1 1}(e_{11}, \ldots, e_{nn})x_{1k_1} \ldots p_{k_n n}(e_{11}, \ldots, e_{nn})x_{nk_n},$$

где $k_1, \ldots, k_n \in \{1, 2, \ldots, n\}$.

Заметим, что коэффициент при $x_{1k_1} \ldots x_{nk_n}$ равен

$$c_{k_1,k_2,\ldots,k_n} = \sum_{e\in\{0,1\}^{n^2}} p_{k_1 1}(e_{11}, \ldots, e_{nn}) \ldots p_{k_n n}(e_{11}, \ldots, e_{nn}).$$

Перманент, по определению, — это сумма слагаемых вида $x_{1k_1} \ldots x_{nk_n}$ по всем попарно различным $k_1, \ldots, k_n$. Таким образом, нам достаточно доказать, что $c_{k_1,k_2,\ldots,k_n} = 1$, если все $k_1, \ldots, k_n$ различны, и $c_{k_1,k_2,\ldots,k_n} = 0$, если среди $k_1, \ldots, k_n$ есть равные. Легко проверить, что если все $k_1, \ldots, k_n$ различны, то

$$p_{k_1 1}(e_{11}, \ldots, e_{nn}) \ldots p_{k_n n}(e_{11}, \ldots, e_{nn}) = 1$$

ровно для одного набора из n^2 нулей и единиц $e = (e_{11}, \ldots, e_{nn})$, а для всех других наборов это произведение равно нулю.

Пусть теперь не все $k_1, \ldots, k_n$ различны, например $k_i = k_j = k$ для каких-то $i \neq j$. Опять же, легко проверить по определению, что всегда

$$p_{ki}(e_{11}, \ldots, e_{nn}) p_{kj}(e_{11}, \ldots, e_{nn}) = 0.$$

Это означает, что произведение

$$p_{k_1 1}(e_{11}, \ldots, e_{nn}) \ldots p_{k_n n}(e_{11}, \ldots, e_{nn})$$

равно нулю для всех наборов из n^2 нулей и единиц e. □

Итак, мы увидели, что задача вычисления определителя лежит в классе VP, а задача вычисления перманента — в классе VNP. Очевидно, что VP ⊂ VNP. Гипотеза Вэлиента состоит в том, что эти классы не совпадают. Неизвестно, однако, как подступиться к этой гипотезе. Более того, доказательство предложения 5.1 демонстрирует, что если коэффициенты полиномов «заданы» множеством, принадлежащим комбинаторному классу NP (в нашем примере — множеству всех матриц перестановок), то соответствующее семейство «должно» лежать в классе VNP. Это служит дополнительным аргументом в пользу его наименования.

Оказывается, что, как и в классе NP, в классе VNP есть в некотором смысле самые сложные задачи, и задача вычисления перманента — одна из них. Определим строго, что означает, что одно семейство полиномов сложнее другого.

Определение 5.3. Полиномиально ограниченное семейство полиномов $\{f_n(x_1, \ldots, x_{m(n)})\}_{n\in\mathbb{N}}$ *сводится* к полиномиально ограниченному семейству полиномов $\{g_n(y_1, \ldots, y_{l(n)})\}_{n\in\mathbb{N}}$, если существует функция $t:\mathbb{N}\to\mathbb{N}$ такая, что $t(n)$ не превышает некоторого полинома от n и выполнено равенство

$$f_n(x_1, \ldots, x_{m(n)}) = g_{t(n)}(a_1, \ldots, a_{l(t(n))}),$$

где $a_1, \ldots, a_{l(t(n))} \in \mathbb{R}\cup\{x_1, \ldots, x_{m(n)}\}$.

Если семейство $\{f_n\}$ сводится к $\{g_n\}$, то пишут:

$$\{f_n\}_{n\in\mathbb{N}} \leqslant \{g_n\}_{n\in\mathbb{N}}.$$

Что́ в сущности означает, что одно семейство полиномов сводится к другому? Это означает, что любой полином из первого семейства можно представить в виде полинома из второго семейства примерно того же размера, в котором вместо переменных подставлены какие-то константы и переменные, от которых зависит полином из первого семейства. Это позволяет говорить, что если полиномы из второго семейства можно вычислять за полиномиальное число умножений, то то же верно и для полиномов первого семейства:

Предложение 5.2. *Если известно, что* $\{f_n\}_{n\in\mathbb{N}} \leqslant \{g_n\}_{n\in\mathbb{N}}$ *и* $\{g_n\}_{n\in\mathbb{N}} \in \mathrm{VP}$, *то и* $\{f_n\}_{n\in\mathbb{N}} \in \mathrm{VP}$.

Семейство полиномов $\{g_n\}_{n\in\mathbb{N}}$ называется *полным* в некотором множестве семейств полиномов A, если для всякого семейства $\{f_n\}_{n\in\mathbb{N}} \in A$ выполнено равенство

$$\{f_n\}_{n\in\mathbb{N}} \leqslant \{g_n\}_{n\in\mathbb{N}}.$$

Из теоремы 5.1 вытекает, что семейство $\{\det_n\}_{n\in\mathbb{N}}$ полно в классе VP. Оказывается, что семейство $\{\mathrm{perm}_n\}_{n\in\mathbb{N}}$ является полным в классе VNP. Это по сути означает, что задача вычисления перманента должна быть очень сложной. Если бы мы умели вычислять перманент матрицы размера $n\times n$ за полиномиальное от n число умножений, то из предложения 5.2 вытекало бы, что $\mathrm{VP}=\mathrm{VNP}$, что крайне неправдоподобно.

* * *

Автор глубоко признателен А. Н. Козачинскому и Н. Ю. Медведю за усилия, затраченные на расшифровку лекции и превращение ее в качественный связный текст.

Приложение. Необходимые сведения

A. Полиномы. Функция $f:\mathbb{R}\to\mathbb{R}$ называется *полиномом от одной переменной*, если ее можно представить в виде

$$f(x)=a_0+a_1x+\ldots+a_nx^n, \tag{1}$$

где $a_0, a_1, \ldots, a_n\in\mathbb{R}$, $a_n\neq 0$. Если f представлен в виде (1), то говорят, что *степень полинома* f равна n.

Важно понимать, что представление в виде (1) единственно. Это означает, что если равенство

$$a_0+a_1x+\ldots+a_nx^n=b_0+b_1x+\ldots+b_mx^m$$

выполнено для всех $x\in\mathbb{R}$, где $a_0, \ldots, a_n, b_0, \ldots, b_m\in\mathbb{R}$, $a_n\neq 0$, $b_m\neq 0$, то

$$n=m \quad \text{и} \quad a_0=b_0,\ a_1=b_1,\ \ldots,\ a_n=b_n.$$

Мы постоянно неявно используем следующую теорему:

Теорема A.1 (Безу). *Отличный от нуля полином $f(x)$ степени n от одной переменной имеет не более n различных корней, т. е.*

$$\bigl|\{x\in\mathbb{R}\mid f(x)=0\}\bigr|\leqslant n.$$

Множество всех полиномов от одной переменной обозначается через $\mathbb{R}[x]$. Мы будем также рассматривать *полиномы от многих переменных*. Пусть зафиксированы переменные $x_1, \ldots, x_k$. *Мономом* называется выражение вида:

$$x_1^{j_1}x_2^{j_2}\ldots x_k^{j_k}, \tag{2}$$

где $j_1, \ldots, j_k$ — неотрицательные целые числа. *Степень монома* (2) по определению полагается равной $j_1+\ldots+j_k$.

Полином от переменных $x_1, \ldots, x_k$ — это сумма конечного числа мономов, взятых с ненулевыми коэффициентами. *Степенью полинома от многих переменных* называется максимум степеней входящих в него мономов. Множество всех полиномов от переменных $x_1, \ldots, x_k$, по аналогии с полиномами от одной переменной, обозначается через $\mathbb{R}[x_1, \ldots, x_k]$.

B. Перестановки и определитель. Функция

$$\sigma:\{1, 2, \ldots, n\}\to\{1, 2, \ldots, n\}$$

называется *перестановкой множества из n элементов*, если σ — биекция. Множество всех перестановок из n элементов обозначается через S_n. Легко видеть, что выполняется следующее утверждение.

Предложение B.1. $|S_n| = n!$.

Четность перестановки σ полагается равной четности количества элементов следующего множества:

$$\{i, j \in \{1, \ldots, n\} \mid i < j,\ \sigma(i) > \sigma(j)\}.$$

Например, такая перестановка 3 элементов:

$$\sigma_1 :\ 1 \mapsto 2,\ 2 \mapsto 3,\ 3 \mapsto 1$$

четная, а такая:

$$\sigma_2 :\ 1 \mapsto 2,\ 2 \mapsto 1,\ 3 \mapsto 3$$

нечетная. Вводится следующее обозначение:

$$\operatorname{sgn}(\sigma) = \begin{cases} 1, & \text{если } \sigma \text{ — четная,} \\ -1, & \text{если } \sigma \text{ — нечетная.} \end{cases}$$

Определителем матрицы размера $n \times n$ называется выражение

$$\det \begin{pmatrix} x_{11} & x_{12} & \ldots & x_{1n} \\ \ldots & \ldots & \ldots & \ldots \\ x_{n1} & x_{n2} & \ldots & x_{nn} \end{pmatrix} = \sum_{\sigma \in S_n} \operatorname{sgn}(\sigma) x_{1\sigma(1)} \ldots x_{n\sigma(n)}. \qquad (3)$$

Непосредственной проверкой можно установить справедливость следующих утверждений.

Предложение B.2. 1. *Если мы к одной строке матрицы прибавим поэлементно другую строку, умноженную на какое-то число, то определитель не изменится.*

2. *Если в матрице мы поменяем местами две строки, то определитель поменяет знак.*

Замечание. На самом деле, у определителя матрицы с вещественными элементами есть вполне ясный геометрический смысл. А именно, он равен (ориентированному) объему параллелепипеда с ребрами-векторами, координаты которых выписаны по столбцам этой матрицы.

C. Комплексные корни. Как известно, квадратных корней из единицы ровно два: 1 и -1. Можно дать определения корней из единицы для больших степеней:

Определение. Число $w \in \mathbb{C}$ называется *комплексным корнем степени n из единицы*, где $n \in \mathbb{N}$, если выполнено равенство

$$w^n = 1.$$

Предложение C.1. *Комплексных корней степени n из единицы ровно n.*

Например, комплексными корнями 3-й степени из единицы являются
$$1,\ \cos\left(\frac{2\pi}{3}\right)+i\sin\left(\frac{2\pi}{3}\right),\ \cos\left(\frac{4\pi}{3}\right)+i\sin\left(\frac{4\pi}{3}\right).$$
Нас будут интересовать комплексные корни степени n из единицы, которые не являются корнями из единицы ни для какой меньшей степени.

Определение. Комплексный корень w степени n из единицы называется *первообразным*, если для всякого $j \in \{1, 2, \ldots, n-1\}$ выполнено равенство
$$w^j \neq 1.$$

Предложение C.2. *Для всякого $n \in \mathbb{N}$ первообразные корни степени n из единицы существуют. Более того, их ровно $\varphi(n)$, где φ — функция Эйлера.*

D. Порядок роста. Пусть $f : \mathbb{N} \to \mathbb{R}$, $g : \mathbb{N} \to \mathbb{R}$ — некоторые функции (последовательности действительных чисел). В математике для краткости используются следующие обозначения.

Определение. $f(n) = O(g(n))$, если найдется такая константа $C > 0$, что для всех n выполнено неравенство
$$|f(n)| \leqslant C|g(n)|$$
(в этом случае говорят, что f есть «O большое» от g).

$f(n) = \Omega(g(n))$, если найдется такая константа $c > 0$, что для всех n выполнено неравенство
$$|f(n)| \geqslant c|g(n)|$$
(f есть «Ω большое» от g).

Например:
$$\sin\left(\frac{1}{n}\right) = O\left(\frac{1}{n}\right),$$
$$e^{\frac{1}{n}} - 1 = \Omega\left(\frac{1}{n}\right).$$

Список литературы

[1] *Baur W., Strassen V.* The complexity of partial derivatives // Theoret. Comput. Sci. 1983. V. 22, № 3. P. 317—330.

[2] *Alman J., Vassilevska Williams V.* Limits on all known (and some unknown) approaches to matrix multiplication, to appear in 59th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2018).

[3] *Bluestein L. I.* A linear filtering approach to the computation of the discrete Fourier transform // IEEE Trans. on Audio and Electroacoustics. 1970. V. 18, issue 4. P. 451—455.

[4] *Cohn H., Kleinberg R., Szegedy B., Umans C.* Group-theoretic algorithms for matrix multiplication // Proceedings of the 46th Annual IEEE Symposium on Foundations of Computer Science. 2005. P. 379—388.

[5] *Cohn H., Umans C.* Fast matrix multiplication using coherent configurations // Proceedings of the Twenty-Fourth Annual ACM—SIAM Symposium on Discrete Algorithms. Philadelphia, PA: SIAM, 2012. P. 1074—1087.

[6] *Cooley J. W., Tukey J. W.* An algorithm for the machine calculation of complex Fourier series // Math. Comp. 1965. V. 19. P. 297—301.

[7] *Hartmann W., Schuster P.* Multiplicative complexity of some rational functions // Theoret. Comput. Sci. 1980. V. 10, № 1. P. 53—61.

[8] *van Leeuwen J., van Emde Boas P.* Some elementary proofs of lower bounds in complexity theory // Linear Algebra and Appl. 1978 V. 19, № 1. P. 63—80.

[9] *Morgenstern J.* Note on a lower bound of the linear complexity of the fast Fourier transform // J. Assoc. Comput. Mach. 1973. V. 20. P. 305—306.

[10] *Ostrowski A. M.* On two problems in abstract algebra connected with Horner's rule // Studies in Mathematics and Mechanics presented to Richard von Mises. New York: Academic Press Inc., 1954. P. 40—48.

[11] *Paterson M. S., Stockmeyer L.* Bounds of evaluation time for rational polynomials // Conference Record 12th Annual Symposium on Switching and Automata Theory. IEEE, 1971. P. 140—143.

[12] *Rader C. M.* Discrete Fourier transforms when the number of data samples is prime // Proc. IEEE. 1968. V. 56, issue 6. P. 1107—1108.

[13] *Scholz A.* Aufgabe 253 // Jahresber. Deutsch. Math.-Verein. 1937. V. 47. P. 41—42.

[14] *Strassen V.* Evaluation of rational functions // Complexity of Computer Computations (Proc. Sympos., IBM Thomas J. Watson Res. Center, Yorktown Heights, N. Y., 1972). New York: Plenum, 1972. P. 1—10, 187—212.

[15] *Strassen V.* Die Berechnungskomplexität von elementarsymmetrischen Functionen und von Interpolationskoeffizienten // Numer. Math. 1972/73. V. 20. P. 238—251.

[16] *Strassen V.* Vermeidung von Divisionen // Crelles J. Reine Angew. Math. 1973. V. 264. P. 184—202.

[17] *Strassen V.* Algebraic Complexity Theory // Handbook of theoretical computer science, Vol. A. Amsterdam: Elsevier, 1990. P. 633—672.

[18] *Williams V. V.* Multiplying matrices faster than Coppersmith—Winograd // STOC'12 — Proceedings of the 2012 ACM Symposium on Theory of Computing. New York: ACM, 2012. P. 887—898.

[19] *Winograd S.* On the number of multiplications necessary to compute certain functions // Comm. Pure Appl. Math. 1970. V. 23. P. 165—179.

[20] *Winograd S.* On computing the discrete Fourier transform // Proc. Nat. Acad. Sci. USA. 1976. V. 73, № 4. P. 1005—1006.

[21] *Пан В. Я.* О способах вычисления значений многочленов // Успехи математических наук. 1966. Т. 21, вып. 1 (127). С. 103—134.

Содержание

Магазин «Математическая книга»

Книги издательства МЦНМО можно приобрести в магазине «Математическая книга» в Москве по адресу: Б. Власьевский пер., д. 11; тел. (495) 745-80-31; `biblio.mccme.ru`

Книга — почтой: `biblio.mccme.ru/shop/order`

Книги в электронном виде: `www.litres.ru/mcnmo`

Мы сотрудничаем с интернет-магазинами

- Книготорговая компания «Абрис»; тел. (495) 229-67-59, (812) 327-04-50; `www.umlit.ru`, `www.textbook.ru`, `абрис.рф`
- Интернет-магазин «Книга.ру»; тел. (495) 744-09-09; `www.kniga.ru`

Наши партнеры в Москве и Подмосковье

- Московский Дом Книги и его филиалы (работает интернет-магазин); тел. (495) 789-35-91; `www.mdk-arbat.ru`
- Магазин «Молодая Гвардия» (работает интернет-магазин): ул. Б. Полянка, д. 28; тел. (499) 238-50-01, (495) 780-33-70; `www.bookmg.ru`
- Магазин «Библио-Глобус» (работает интернет-магазин): ул. Мясницкая, д. 6/3, стр. 1; тел. (495) 781-19-00; `www.biblio-globus.ru`
- Спорткомплекс «Олимпийский», 5-й этаж, точка 62; тел. (903) 970-34-46
- Сеть киосков «Аргумент» в МГУ; тел. (495) 939-21-76, (495) 939-22-06; `www.arg.ru`
- Сеть магазинов «Мир школьника» (работает интернет-магазин); тел. (495) 715-31-36, (495) 715-59-63, (499) 182-67-07, (499) 179-57-17; `www.uchebnik.com`
- Сеть магазинов «Шаг к пятерке»; тел. (495) 728-33-09, (495) 346-00-10; `www.shkolkniga.ru`
- Издательская группа URSS, Нахимовский проспект, д. 56, Выставочный зал «Науку — Всем», тел. (499) 724-25-45, `www.urss.ru`
- Книжный магазин издательского дома «Интеллект» в г. Долгопрудный: МФТИ (новый корпус); тел. (495) 408-73-55

Наши партнеры в Санкт-Петербурге

- Санкт-Петербургский Дом книги: Невский пр-т, д. 62; тел. (812) 314-58-88
- Магазин «Мир науки и медицины»: Литейный пр-т, д. 64; тел. (812) 273-50-12
- Магазин «Новая техническая книга»: Измайловский пр-т, д. 29; тел. (812) 251-41-10
- Информационно-книготорговый центр «Академическая литература»: Васильевский остров, Менделеевская линия, д. 5
- Киоск в здании физического факультета СПбГУ в Петергофе; тел. (812) 328-96-91, (812) 329-24-70, (812) 329-24-71
- Издательство «Петроглиф»: Фарфоровская, 18, к. 1; тел. (812) 560-05-98, (812) 943-80-76; `k_i_@bk.ru`
- Сеть магазинов «Учебная литература»; тел. (812) 746-82-42, тел. (812) 764-94-88, тел. (812) 235-73-88 (доб. 223)

Наши партнеры в Челябинске

- Магазин «Библио-Глобус», ул. Молдавская, д. 16, `www.biblio-globus.ru`

Наши партнеры в Украине

- Александр Елисаветский. Рассылка книг наложенным платежом по Украине: тел. 067-136-37-35; `df-al-el@bk.ru`