Стивен Смейл

О проблемах вычислительной сложности

Мы сейчас обсудим одну задачу, которая в элементарной форме иллюстрирует основные трудности теории вычислительной сложности. Для полинома $f \in \mathbb{Z}[t]$ определим число $\tau(f)$ следующим образом. Рассмотрим последовательность $(1, t, u_1, \ldots, u_m = f)$, в которой каждый последующий член получается из некоторых двух предшествующих: $u_k = u_i \circ u_j$, $i, j < k$; под операцией $\circ$ здесь подразумевается одна из трех арифметических операций (сложение, вычитание, умножение). Инвариант $\tau(f)$ равен наименьшему возможному m.

Имеется следующая гипотеза Шуба—Смейла: *количество различных целых корней многочлена f не превосходит $\tau(f)^c$, где c — некоторая абсолютная константа.*

Пример. *Последовательность $1, t, t^2, t^{2^2}, \ldots, t^{2^k}$, $t^{2^k} - 1$ показывает, что $\tau(t^{2^k} - 1) \leqslant k + 1$. Но при этом количество различных корней многочлена $t^{2^k} - 1$ равно 2^k. Поэтому для различных комплексных корней аналогичная гипотеза неверна.*

Аналогичный пример можно построить с помощью многочленов Чебышева. Многочлены Чебышева вычисляются с помощью простой рекуррентной формулы. Они тоже дают пример многочленов высокой степени с малым τ. При этом все корни многочленов Чебышева вещественны и попарно различны. Для различных вещественных корней аналогичная гипотеза тоже неверна.

Стивен Смейл (Stephen Smale), профессор Калифорнийского Университета (США).

Теорема 1 (Шуб—Смейл). *Из гипотезы Шуба—Смейла следует, что* $\mathrm{P} \neq \mathrm{NP}/_{\mathbb{C}}$.

Теперь нужно объяснить, что означает $\mathrm{P} \neq \mathrm{NP}/_{\mathbb{C}}$.

Прежде всего отметим, что у алгебраистов нетривиальные проблемы обычно начинаются с диофантовых уравнений, соответствующих алгебраическим кривым, т. е. двум переменным. У нас проблемы начинаются уже в случае одной переменной.

Если забыть про $\mathbb{C}$, то проблема $\mathrm{P} \neq \mathrm{NP}$ — это одна из ключевых проблем компьютерной математики. Вместе с гипотезой Пуанкаре и гипотезой о нулях дзета-функции Римана она является одной из важнейших проблем математики — это подарок от computer science.

Рассмотрим многочлены $f_1(z_1, \ldots, z_n), \ldots, f_k(z_1, \ldots, z_n)$ над $\mathbb{C}$. Спрашивается, имеют ли эти многочлены общий нуль? Это — задача распознавания свойства: в качестве условия задаются многочлены $f_1, \ldots, f_k$ (точнее говоря, задается несколько комплексных чисел — коэффициентов многочленов), а результатом работы должен быть один из двух ответов: «да» (есть общий нуль) или «нет» (общего нуля нет).

Теорема Гильберта о нулях дает следующий ответ: общего нуля не существует тогда и только тогда, когда существуют такие многочлены $g_1, \ldots, g_k$, что $\sum g_i f_i = 1$.

Теорема Гильберта о нулях — критерий, но не метод. Она не дает никакого алгоритма. Но примерно 10 лет назад Браунвелл[1)] показал, что в теореме Гильберта о нулях можно считать, что

$$\deg g_i \leqslant \max(3, \max \deg f_i)^n,$$

причем этот результат неулучшаем.

Теорема Браунвелла дает алгоритм: все сводится к решению системы линейных уравнений для коэффициентов многочленов g_i.

Займемся теперь вопросом о скорости этого алгоритма: сколько арифметических операций нужно выполнить, чтобы ответить на поставленный вопрос. Назовем *размером* входных данных количество коэффициентов многочленов f_i, а *временем* работы алгоритма назовем количество арифметических операций. Будем называть данный алгоритм *алгоритмом с полиномиальным временем*, если

$$\text{время} \leqslant (\text{размер})^C, \tag{1}$$

где C — некоторая константа.

[1)] W. Brownawell. Bounds for the degrees in the Nullstellensatz // Annals of Math. **126** (1987), № 3. P. 577–591.

Алгоритмы с полиномиальным временем — это как раз те алгоритмы, которые имеет смысл практически реализовывать на вычислительной машине. Если, скажем, время зависит от размера экспоненциально, то при увеличении размера входных данных время быстро выходит за разумные пределы. Алгоритм Браунвелла является алгоритмом с экспоненциальным временем. Экспоненциальная верхняя оценка для этого алгоритма легко выводится, например, из гауссова метода исключения для решения системы линейных уравнений.

Гипотеза такова: задача $HN/_{\mathbb{C}}$ (существует ли общий нуль системы полиномиальных уравнений над $\mathbb{C}$) трудноразрешима, т.е. не существует алгоритма с полиномиальным временем для решения этой задачи.

Здесь имеется в виду алгоритм не в смысле машины Тьюринга, а алгоритм над $\mathbb{C}$ в следующем смысле. Алгоритм — это ориентированный граф с одной вершиной, в которую не ведет ни одного ребра (*входом*). Граф может иметь циклы. Он задает работу вычислительной машины следующим образом. На вход подается бесконечная в обе стороны последовательность комплексных чисел $(\dots, 0, z_1, \dots, z_n, 0, \dots)$, среди которых только $z_1, \dots, z_n$ отличны от нуля, никаких ограничений на величину n не предполагается, так что такая модель вычислительной машины может работать со сколь угодно длинными последовательностями чисел. Вершины этого графа относятся к одному из трех типов:

- *Выходы.* Из них не ведет ни одного ребра. По достижении такой вершины работа заканчивается.

- *Вычислительный узел.* В вычислительный узел входит одно ребро и из него выходит тоже одно ребро. В вычислительном узле производится арифметическая операция с какими-то членами последовательности и один из членов последовательности заменяется на результат вычислений. Кроме того, можно все члены последовательности умножить на одно и то же число или произвести сдвиг последовательности.

- *Узел ветвления.* В узел ветвления входит одно ребро, а выходят из него два ребра, на которых стоят пометки «да» и «нет». В узле ветвления выясняется, верно ли что $z_i = 0$. Если $z_i = 0$, то мы идем дальше по ребру с пометкой «да», а если $z_i \neq 0$, то мы идем дальше по ребру с пометкой «нет». (Для вычислений над $\mathbb{R}$ вместо этого можно ввести проверку типа $x_i > 0$ или $x_i \geqslant 0$.)

На выходе алгоритма тоже получается последовательность чисел. В интересующем нас алгоритме $HN/_{\mathbb{C}}$ на выходе только один ненулевой элемент, который может принимать ровно два значения, соответствующие ответам «да» и «нет».

Такое определение алгоритма было дано Л. Блюм, С. Смейлом и М. Шубом в конце 80-х годов. Странно, что раньше никто не додумался до этого совершенно естественного определения. Подробно ознакомиться с теорией таких алгоритмов можно по книге L. Blum, F. Cucker, M. Shub., S. Smale. Complexity and Real Computation. Springer Verlag, 1997.

С описанным выше алгоритмом естественным образом связана функция «входа—выхода». Она определена на некотором множестве входных данных (например, машина не может производить деление на нуль, поэтому при некоторых входных данных она останавливается).

Размером входных данных назовем число n, а временем работы при данном входе — длину пути от входа к выходу (на разных входах эти пути могут быть различными). Алгоритмы с полиномиальным временем удовлетворяют неравенству (1) для некоторой константы C при всех входах. Класс таких алгоритмов обозначается $\mathrm{P}/_{\mathbb{C}}$.

После этого определения вопрос о том, существует ли полиномиальный алгоритм для задачи $HN/_{\mathbb{C}}$, приобретает строгий математический смысл. Заметим, что утверждение о том, что такого алгоритма не существует, в точности эквивалентно утверждению $\mathrm{P} \neq \mathrm{NP}/_{\mathbb{C}}$ (определение $\mathrm{NP}/_{\mathbb{C}}$ пока не давалось и на этой лекции не будет дано).

Вместо поля $\mathbb{C}$ можно взять произвольное поле K и определить вычислительную машину над произвольным полем. Например, поле $K=\mathbb{Z}_2$ соответствует определению алгоритма, принятому в логике и computer science.

Можно также поставить вопрос об общих нулях многочленов над $\mathbb{Z}_2$. Гипотеза о том, что не существует полиномиального алгоритма для решения этой задачи, эквивалентна гипотезе $\mathrm{P} \neq \mathrm{NP}$ в ее классическом варианте.

Для числа $m \in \mathbb{Z}$ можно определить инвариант $\tau(m)$ по аналогии с инвариантом τ для многочленов. А именно, рассмотрим аналогичную последовательность $(1, m_1, \ldots, m_k = m)$ и определим $\tau(m)$ как минимальное возможное k. С помощью формулы Стирлинга можно доказать, что $\tau(m!) \leqslant (\ln m)^C$. Есть предположение, что верна и противоположная оценка такого же вида: $(\ln m)^{C'} \leqslant \tau(m!)$; эта проблема связана с разложением на простые множители.

На первый взгляд эти две проблемы (об инварианте τ для полиномов и для чисел) друг с другом не связаны.

Вернемся к проблеме $\mathrm{P} \neq \mathrm{NP}/_K$. Мы не будем определять, что такое $\mathrm{NP}/_K$, вместо этого будем говорить об эквивалентной проблеме Гильберта о нулях над произвольным полем K: $HN/_K \notin \mathrm{P}/_K$. (Если поле не алгебраически замкнуто, то теорема Гильберта о нулях неверна, но задача об общих нулях системы многочленов имеет смысл над произвольным полем; здесь имеется в виду именно эта задача.)

В случае не алгебраически замкнутого поля доказано следующее утверждение.

Теорема 2. *Если поле K не алгебраически замкнуто и* $\operatorname{char} K = 0$*, то* $\mathrm{P} \neq \mathrm{NP}/_K$.

Для поля $\mathbb{Z}_2$, которое тоже не алгебраически замкнуто, вопрос остается открытым (характеристика этого поля отлична от нуля).

Вернемся к алгебраически замкнутым полям. Для алгебраически замкнутого поля K ($\operatorname{char} K = 0$) проблема $\mathrm{P} \neq \mathrm{NP}/_K$ эквивалентна проблеме $\mathrm{P} \neq \mathrm{NP}/_{\mathbb{C}}$. Поэтому все сводится к одному полю, например, полю $\mathbb{C}$ или полю $\overline{\mathbb{Q}}$ (так обозначается алгебраическое замыкание поля $\mathbb{Q}$). Это — один из основных результатов упомянутой выше книги. Доказательство использует понятие высоты алгебраического числа.

Представляется весьма правдоподобным, что $\mathrm{P}/_K = \mathrm{P}/_{\mathbb{F}_2}$ для любого конечного поля K. Но для полей конечной характеристики вопросов больше, чем ответов.

Рассмотрим вопрос об эквивалентности проблем над $\mathbb{C}$ и над $\overline{\mathbb{Q}}$. Одна из основных проблем при переходе от комплексных чисел к алгебраическим связана с тем, что нужно избавиться от комплексных констант, которые могут использоваться при вычислениях. Вообще говоря, они могли бы сильно упростить вычисления. Однако доказано, что такого упрощения не происходит.

В упомянутой выше книге не рассматривался вопрос о связи проблем $\mathrm{P} \neq \mathrm{NP}/_{\mathbb{Z}_2}$ и $\mathrm{P} \neq \mathrm{NP}/_{\mathbb{C}}$. Именно первая из них относится к классической computer science. В предисловии к этой книге Дик Карп высказал предположение, что эти проблемы никак друг с другом не связаны. Но уже после того как книга была написана, Смейл заметил следующее.

Напомним, что интерес к алгоритмам с полиномиальным временем связан с тем, что именно их можно эффективно реализовывать на компьютерах. Но сейчас используется еще один важный класс алгоритмов — так называемые BPP-алгоритмы. В этих алгоритмах разрешается «подбрасывать монетку» и в зависимости от полученного результата производить те или иные вычисления. Требуется, чтобы правильный ответ получался в «квалифицированном большинстве» случаев. Тогда, повторив вычисления много раз, можно получить результат, который будет правильным с очень большой вероятностью. Например, если правильный результат получается с вероятностью 3/4, то после 50 повторений вычислений вероятность ошибки будет равна единице, деленной на число атомов во Вселенной.

С математической точки зрения условие BPP накладывает меньше ограничений, чем условие P, но с практической точки зрения BPP-алгоритмы столь же хороши, как и P-алгоритмы.

Теорема 3 (Смейл). *Если* $\mathrm{BPP} \not\supseteq \mathrm{NP}$*, то* $\mathrm{P} \neq \mathrm{NP}/_{\mathbb{C}}$*.*

С точки зрения современной computer science $\mathrm{BPP} \not\supseteq \mathrm{NP}$ — это нечто очень похожее на $\mathrm{P} \neq \mathrm{NP}$.