Колмогоровская сложность

Ю. Л. Притыкин

Данная заметка представляет собой переработанный конспект лекции (с задачами, решения многих из которых были разобраны на лекции), прочитанной в школе №57 г. Москвы школьникам 11д класса в 2006 году и повторенной школьникам 11в класса в 2008 году. Она основана на находящейся в стадии написания книге по колмогоровской сложности В. А. Успенского, Н. К. Верещагина, А. Шеня [2], которую тем не менее можно рекомендовать для дальнейшего чтения по этой теме, см. файлы по ссылке.

Речь пойдёт о фундаментальном понятии в теории алгоритмов — колмогоровской сложности (другие названия — описательная сложность, дескриптивная сложность, алгоритмическая сложность, колмогоровская энтропия). Его окончательно сформулировал Колмогоров (а также независимо Соломонов и Чейтин) в своих работах 1965 года [1]. Оно формализует интуитивное представление о количестве информации в конечных объектах — в нашем случае это слова из 0 и 1.

Интуитивно ясно, что строку

«01010101010101010101010101010101»

можно описать короче — «15 слов 01», в то время как строку

«11011000011101001101001111001 0»

никак существенно короче не запишешь. Значит, в первой из них информации меньше, чем во второй, хотя они имеют одинаковую длину. Или, например, чтобы задать слово из миллиона нулей, достаточно сказать «1000000 нулей», и вовсе необязательно все эти нули выписывать. Более жизненный пример: часто при работе с файлами большого размера приходится использовать различные архиваторы — `zip`, `gzip`, `rar` и т. д. — для компрессии, то есть получения более короткого описания. Важно, что если потом разархивировать файл, то получится в точности исходный, и никакая информация не потеряется. Только такие способы описания мы и будем рассматривать. Неформально, количеством информации в слове назовём минимальную длину всевозможных его описаний.

Для точного определения способа описания мы воспользуемся понятием алгоритма. Интуитивное представление о том, что такое алгоритм, есть

у каждого (например, программа на почти любом известном вам языке программирования — Pascal, C, C++, Java и т. д.). Мы ограничимся только рассмотрением декомпрессоров, то есть алгоритмов получения по описанию исходного слова (естественно потребовать, чтобы по любому слову декодирование происходило однозначно). *Способом описания* (или *декомпрессором*) будем называть произвольную вычислимую с помощью алгоритма функцию D на множестве A всех двоичных слов (включая пустое слово Λ): $A = \{\Lambda, 0, 1, 00, 01, 10, 11, 000, \dots\}$. Значения функции D тоже лежат в A. Однако это ограничение можно обойти и распространить D на многие другие конечные объекты. Для этого можно рассматривать какое-нибудь их кодирование двоичными словами. Мы разрешаем вычислимым функциям быть неопределёнными на некоторых словах (возможно, даже на всех) — содержательно это означает, что соответствующая программа не завершает работу (зацикливается) на некоторых входах.

Через $|x|$ обозначим длину слова $x \in A$.

Определение 1. Назовём число

$$K_D(x) = \min\{|y| : D(y) = x\}$$

сложностью слова x *относительно способа описания* D. (Мы считаем $\min \varnothing = +\infty$.)

Задача 1. Какую функцию сложности K_D определяют следующие способы описания? а) $D(y) = \Lambda$ для любого слова y из A; б) $D(\mathrm{bin}(n)) = \underbrace{00\dots0}_{n}$ (здесь $\mathrm{bin}(n)$ — двоичная запись целого положительного числа n); в) $D(y) = y$ для любого слова y из A.

Определение сложности очень сильно зависит от способа описания. Подбирая такой способ, мы можем сделать очень маленькой сложность любого конкретного слова или даже семейства слов. Наша задача — выбрать оптимальный способ, который хорош для всех слов одновременно. Естественно считать, что способ описания тем лучше, чем более короткие описания он даёт.

Определение 2. Способ описания D_1 *не хуже* способа D_2, если для некоторой константы C и для всех слов x имеем

$$K_{D_1}(x) \leqslant K_{D_2}(x) + C.$$

Мы будем пренебрегать различием на константу. Без этого дальнейшую теорию построить не удаётся (в частности, для аналогичного определения без константы неверна теорема Соломонова – Колмогорова о существовании оптимального способа описания, см. далее).

Прежде чем строить наилучший способ описания, начнём с малого. Пусть есть два способа описания D_1 и D_2. Как построить способ D, который не хуже каждого из них? Положим

$$D(0y) = D_1(y),\ D(1y) = D_2(y).$$

Таким образом, учитываются сразу оба способа описания. Чтобы потом можно было декодировать полученное описание, приписываем к нему информацию о том, каким способом оно было получено — первым или вторым.

Задача 2. Проверьте, что $K_D(x) \leqslant K_{D_1}(x) + 1$ и $K_D(x) \leqslant K_{D_2}(x) + 1$ для любого x.

Обобщив эту идею, мы можем построить такой оптимальный способ описания D, что для любого другого способа D' существует такая константа $C_{D'}$, что для любого слова x имеем

$$K_D(x) \leqslant K_{D'}(x) + C_{D'}$$

(это утверждение называется теоремой Соломонова – Колмогорова). Действительно, мы можем учитывать все имеющиеся способы записи одновременно, просто указывая в начале описания, какой способ использовался. Положим

$$D(py) = p(y),$$

где p — произвольный алгоритм (записанный двоичным кодом). Ясно тогда, что D хуже способа описания, задаваемого программой p, не более чем на константу — длину текста программы p.

Таким образом, когда D подают на вход текст, он выделяет из его начала текст программы, а потом применяет эту программу к оставшемуся тексту. Однако возникает проблема. Пусть, например, нам дают слово 01. Мы не знаем, применять ли программу 0 к слову 1 или программу 01 к пустому слову (или, может быть, программу с пустым текстом Λ к слову 01). Поэтому мы должны придумать такое описание пары p, y, чтобы однозначно выделять из него p. Это несложно: например, будем удваивать каждый символ слова p, а по его окончании вставим 01. Например, если $p = 011001$ и $y = 01001$, то получим код 0011110000110101001.

Задача 3. а) Описанный выше способ кодирования пары слов x и y одним словом даёт оценку $2|x| + |y| + 2$ на длину кода. Как можно было бы эту оценку улучшить (например, получить оценку $|x| + |y| + 2\log|x| + 2$ или $|x| + |y| + \log|x| + 2\log\log|x| + 2$, и т. д.)[1)]? б) Можно ли придумать способ кодировать пару слов x, y словом длины $|x| + |y| + c$, где c — константа?

Оптимальный способ описания построен.

[1)] Всюду под log подразумевается $\log_2$.

Определение 3. Фиксируем некоторый оптимальный способ описания D. Назовём

$$K(x) = \min\{|y| : D(y) = x\}$$

колмогоровской сложностью слова x.

Задача 4. Докажите, что замена оптимального способа описания в этом определении на другой (оптимальных может быть много) приводит к изменению функции сложности не более чем на константу.

Задача 5. Докажите, что колмогоровская сложность любого слова конечна.

В силу произвола в выборе оптимального способа описания в определении все утверждения про колмогоровскую сложность носят асимптотический характер. Рассмотрим пример: как может меняться сложность слова при приписывании к нему символа 1? Ясно, что от этого количество информации в слове меняется несущественно. Можно доказать, что $K(1x) = K(x) + O(1)$. (Здесь $O(1)$ обозначает функцию от x, ограниченную константой. Далее мы будем использовать общее определение $O(f(x))$ как такой функции $g(x)$, что для некоторой константы C и всех x, больших некоторого D, выполняется неравенство $g(x) \leqslant Cf(x)$ (будем для удобства считать, что все рассматриваемые функции неотрицательные).) Неформально, для этого достаточно показать, что можно по каждому из слов x и $1x$ легко получать другое, не используя дополнительной информации.

Задача 6. а) Проведите последнее рассуждение формально. б) Докажите, что $K(xx) = K(x) + O(1)$. в) Докажите, что $K(\underbrace{00\ldots0}_{n}) \leqslant \log n + O(1)$.

Задача 7. Докажите, что $K(x) \leqslant |x| + O(1)$.

Задача 8. Пусть P — произвольный алгоритм. Докажите, что $K(P(x)) \leqslant K(x) + O(1)$.

Задача 9. а) Покажите, что если описания рассматривать не в бинарном алфавите, а в алфавите из 4 символов, то сложность уменьшится вдвое. б) Сформулируйте и докажите аналогичное утверждение для перехода к произвольному конечному алфавиту.

Оказывается, что неравенство в задаче 7 для большинства слов близко к равенству. Действительно, давайте посчитаем, сколько слов может иметь сложность меньше некоторого фиксированного числа n. Слов сложности 0 не более одного — оно должно иметь описание Λ, слов сложности 1 не более двух — их описания могут быть только 0 и 1, и так далее. Поэтому слов сложности меньше n не больше, чем $1 + 2 + 4 + \cdots + 2^{n-1} = 2^n - 1$. Таким образом, для любого натурального n существует менее 2^n слов x, для которых $K(x) < n$. Отсюда легко получаем, что доля слов сложности

менее $n-c$ среди всех слов длины n меньше $\frac{2^{n-c}}{2^n} = 2^{-c}$. Например, доля слов сложности меньше 90 среди слов длины 100 меньше $\frac{1}{1024}$. Таким образом, большинство слов несжимаемы или почти несжимаемы.

Удивительно, что в полученном утверждении — существует менее 2^n слов сложности меньше n — нет никаких констант.

Задача 10. Где в этом утверждении зависимость от фиксированного в определении колмогоровской сложности оптимального способа описания?

Задача 11. Покажите, что для некоторого фиксированного c и всех n количество слов сложности меньше n заключено между 2^{n-c} и 2^n.

Задача 12. Покажите, что среднее арифметическое сложностей всех слов длины n равно $n + O(1)$.

Задача 13. а) Докажите, что если y получено из слова x длины n заменой одного символа, то $K(y) = K(x) + O(\log n)$. б) Может ли тем не менее $K(y)$ существенно отличаться от $K(x)$ (например, в 1000 раз)?

Задача 14. Докажите, что $K(xy) \leqslant K(x) + K(y) + 2\log K(x) + O(1)$.

Задача 15. а) Докажите, что $K(x, y) \leqslant K(x) + K(y) + O(\log n)$ для любых x и y длины не более n (здесь под парой слов x, y понимается её кодирование двоичным словом в некоторой раз навсегда выбранной стандартной кодировке). б) Можно ли доказать неравенство $K(x, y) \leqslant K(x) + K(y) + O(1)$? (Указание: можно ли доказать $K(x, y) \leqslant |x| + |y| + O(1)$?) в) Как соотносятся между собой $K(x, y)$ и $K(xy)$?

Задача 16. а) Докажите $2K(xyz) \leqslant K(xy) + K(xz) + K(yz) + O(\log n)$ для всех слов x, y, z длины не более n. б) Пусть тело в трёхмерном пространстве имеет объём V, а площади его ортогональных проекций на плоскости xOy, xOz и yOz (в прямоугольной системе координат) равны соответственно S_{xy}, S_{xz} и S_{yz}. Докажите неравенство $V^2 \leqslant S_{xy} \cdot S_{xz} \cdot S_{yz}$.

Было бы очень полезно уметь вычислять $K(x)$. Это можно пытаться делать следующим образом. Возьмём алгоритм оптимальной декомпресии (тот самый, который был зафиксирован в определении колмогоровской сложности). Запустим этот алгоритм на пустом слове. Если получилось x, значит, $K(x) = 0$. Если нет, запустим алгоритм на словах 0 и 1. Если на каком-то из них выдан ответ x, значит, $K(x) = 1$. И так далее: если мы уже знаем, что $K(x) > n - 1$, и нашлось слово длины n, на котором алгоритм выдаёт x, значит, $K(x) = n$. Однако есть проблема: уже на первом шаге, пытаясь применить программу к пустому слову, мы можем столкнуться с трудностью — алгоритм может не завершить работу. И до всего следующего дело уже не дойдёт.

Задача 17. Покажите, что оптимальный способ описания — не всюду определённая функция.

Утверждение этой задачи может показаться странным, ведь если способ описания определён не всюду, мы можем доопределить его в некоторых точках — ясно, что от этого он может только улучшиться. Однако формального противоречия здесь нет (только философское) — чтобы функция осталась вычислимой, её можно доопределить лишь в конечном количестве точек.

В заключение приведём любопытное рассуждение, показывающее, что вычислять функцию $K(x)$ не удаётся (ни вышеприведённым способом, ни каким-либо другим).

Пусть алгоритмически вычислимая функция f оценивает снизу колмогоровскую сложность, то есть $f(x) \leqslant K(x)$ для любого x. Покажем, что такая оценка может быть только тривиальной: для некоторого C и для всех x выполнено $f(x) \leqslant C$. Предположим обратное — пусть f не ограничена. Пользуясь этим, построим следующий алгоритм P. Получив на вход натуральное число n, он запускает одновременно алгоритм для вычисления f на всех словах (отдельный вопрос — как запускать одновременно счётное число программ; обдумайте этот вопрос самостоятельно). Время от времени на каких-то из слов алгоритм выдаёт ответ, и мы можем проверить, верно ли для этого слова x, что $f(x) \geqslant n$. Поскольку по предположению f не ограничена, обязательно когда-нибудь найдётся слово, для которого это неравенство действительно выполнено. Первое найденное такое слово алгоритм P и выдаёт в качестве ответа.

Имеем $f(P(n)) \geqslant n$, а значит, и $K(P(n)) \geqslant n$. С другой стороны, по задаче 8 имеем $K(P(n)) \leqslant K(n) \leqslant \log n + O(1)$. Неравенство $\log n + O(1) \geqslant n$ нарушается при достаточно больших n — противоречие.

Чтобы доказать невычислимость функции K, осталось заметить, что она является собственной оценкой снизу.

Задача 18. Докажите, что функция, равная на слове x его кратчайшему описанию (при оптимальном способе описания), не вычислима алгоритмически. (Во многом именно поэтому мы ограничиваемся рассмотрением декомпрессоров.)

Задача 19. Существует ли алгоритм, которому можно на вход подать тексты двух программ, задающих способы описания D_1 и D_2, про которые известно, что D_1 не хуже D_2, и этот алгоритм выдаст константу в неравенстве $K_{D_1}(x) \leqslant K_{D_2}(x) + C$ из определения?

Задача 20. Пусть $B(n) = \min\{N \in \mathbb{Z} : K(m) > n \text{ для любого } m > N\}$ (регулятор сходимости $K(m)$ к ∞). Тогда для любой алгоритмически вычислимой функции $f\colon \mathbb{N} \to \mathbb{N}$ почти для всех n верно $B(n) \geqslant f(n)$ (то есть B растёт быстрее любой вычислимой функции!).

Задача 21. Пусть $0 < \alpha < 1$. а) Докажите, что существуют сколь угодно длинные слова, у которых каждое начало x имеет сложность

$K(x) > \alpha|x| + O(1)$. б) Докажите, что существуют сколь угодно длинные слова, у которых каждое подслово x сложно: $K(x) > \alpha|x| + O(1)$.

Список литературы

[1] А. Н. Колмогоров. *Три подхода к определению понятия «количество информации»* // Проблемы передачи информации, т. 1, №1, 1965. С. 3–11.

[2] В. А. Успенский, Н. К. Верещагин, А. Шень. *Колмогоровская сложность*. Неопубликованная книга, см.: `http://kolmsem.math.ru/rus/materials/materials.html`

Ю. Л. Притыкин: МГУ им. М. В. Ломоносова,
email: `yura@mccme.ru`