Лекции по математической логике и теории алгоритмов
А. Л. Семенов, 2022
Лекция 14

1 Сложность вычислений и объектов

1.1 Колмогоровская сложность

Начнем с неформального примера.
Представьте себе 1 миллион написанных подряд, через запятую и пробел слов "Да". Длина этой цепочки, видимо, 4 миллиона. Но мы ее описали намного короче, из нашего описания видно, что сложность цепочки не так уж велика.
Как мы уже делали раньше, будем считать, что наши объекты — это двоичные цепочки.

Определение 1. *Способ описания* — любая вычислимая функция. Мы говорим, что она перерабатывает описание в объект с этим описанием.
Сложность объекта x при заданном способе описания A, обозначение — $C_A(x)$ — это минимальная длина описания этого объекта способом A.

Задача 1. Теорема Колмогорова. Существует способ описания $U(p)$, дающий самые короткие описания в следующем смысле:
Для всякого способа описания A существует число D такое, что для любого объекта x

$$C_U(x) < C_A(x) + D$$

Подсказка. Вот основные шаги доказательства:

- Рассмотрите универсальную функцию, которая моделирует любой способ описания.
- Сравните минимальную длину описания при произвольном способе с длиной подходящего описания для универсальной функции.
- Используйте экономный способ кодирования, при котором длина кода пары лишь на константу, зависящую от первого аргумента (программы), больше длины второго аргумента (исходного данного). Код может начинаться с указания длины первого аргумента, при этом должно быть понятно, где это указание закачивается.

Дальнейшее развитие идей Колмогорова о сложности объектов породило алгоритмическую теорию информации. Одним из первых результатов здесь — это ответ на вопрос: как определить случайность бесконечной последовательности?
Ответ: бесконечная последовательность случайна, если ее начальные отрезки нельзя описать на длине существенно меньшей, чем их длина, то есть, если сложность начальных отрезков близка к их длине.

1.2 Сложность вычислений

Начнем с примеров.
Рассмотрим множество натуральных чисел: 23, 11, 44, 29, 18, 32, 19, 35, 67.
и несколько вопросов (задач) об этом множестве:

1. Есть ли среди этих чисел число 54?
2. Есть ли среди этих чисел два числа, сумма которых равна 73?
3. Можно ли среди этих чисел выбрать несколько, сумма которых (взятых по одному разу каждый) равна 100?

Как сравнить эти задачи по сложности вычислений? Такое сравнения полезно, в частности, если мы пишем программу для компьютера и выбираем для нее самый быстрый алгоритм.
Очевидная идея — посчитать количество операций, в нашем случае — сложения и проверки равенства, которые мы должны сделать. Если идти дальше по этому пути, можно получить понятие, так называемой алгебраической сложности, где считается число операций.

Можно рассматривать произвольные задачи и произвольные Действия переработки. Однако имеет смысл некоторым образом ограничить сложность одного шага, иначе за один шаг можно было бы решить всю задачу.
Для того, чтобы ограничить сложность одного шага, можно все алгоритмы реализовать как, например машины Тьюринга.
Для фиксированной машины Тьюринга можно говорить о числе шагов алгоритма. Конечно, число шагов будет зависеть от исходных данных, то есть — это функция от исходного данного. Естественная, хотя и не единственная точка зрения состоит в том, чтобы рассматривать ограничение по числу шагов в зависимости от размера (длины) исходного данного.
Мы получаем следующие определения.

Определение 2. *Время вычисления алгоритма* — длина (число шагов) вычисления. *Время вычисления алгоритма ограничено функцией* f (от натурального аргумента), если для всех исходных данных длины не больше n время вычисления не больше $f(n)$.

Огромный, быстро накопленный опыт построения и использования алгоритмов для реальных компьютеров показал, что класс реально используемых алгоритмов составляют алгоритмы. для которых время вычисления ограничено полиномом от длины исходного данного.

Определение 3. *функция H полиномиально вычислима*, если существует машина Тьюринга, время вычисления для которой ограничено заданным полиномом от длины исходного данного, а вычисляемая функция — это H.

1.3 Переборные задачи

Пример, с которого мы начали лекцию — это задача о рюкзаке. Вот ее точная формулировка:

Определение 4. *Задача о рюкзаке*
Дано: множество натуральных чисел $A = \{a_1, ...a_n\}$ и число b
Требуется узнать: существует ли подмножество множеств A, сумма элементов которого равна b.

Чтобы обсуждать сложность данной задачи, надо договориться о способе кодирования исходных данных с помощью двоичных цепочек. Общий способ может состоять в том, что мы все натуральные числа из A и число b записываем в двоичной системе счисления, затем в двоичной записи все символы удваиваем, потом выписываем все записи подряд, разделяя их символами 01.
Наша задача обладает следующим свойством: если нам даны номера элементов подмножества A, то мы можем за полиномиальное время проверить, что сумма элементов с этими номерами равна b. Таким образом, перебирая всевозможные множества номеров, мы можем решить задачу. Перебор ограничен конечным числом возможностей, но этих возможностей 2^n, где n количество элементов множества A.
Мы видим, что проверка того, что мы нашли решение, занимает полиномиальное время, но поиск решения перебором может потребовать экспоненциального времени. В этом неформальном обсуждении мы не делаем различия между размером исходных данных и количеством элементов в A, это не влияет на качественное различие в полиномиально и экспоненциально сложных вычислениях.

Задача 2. (Д) Предположим, мы можем решать задачу о рюкзаке за полиномиальное время. Как нам тогда за полиномиальное время находить нужное множество номеров?

Определение 5. *Переборная задача*
Дано: Свойство $P(x, y)$ пар двоичных цепочек, вычислимое за полиномиальное время.
Требуется узнать: для всякого существует ли *догадка* y, длина которого не больше длины x для которого выполнено $P(x, y)$.

Ограничение на длину можно ослабить, но это не меняет существа определения.
Вот еще один пример переборной задачи.

Определение 6. *Задача* SAT:

Исходное данное: формула логики высказываний (точнее — ее двоичный код).
Требуется узнать существует ли набор значений для атомных высказываний формулы, обращающих формулу в истину.

Здесь, конечно, исходное данное – это двоичный код формулы, а догадка — это список значений атомов, тоже в форме двоичного кода, из этих двух компонентов образуется код пары.

1.4 Универсальная переборная задача

Замечательный факт, открытый Карпом и Левиным.
Проблема тысячелетия.

Задача 3. Если SAT полиномиально вычислима, то и любая полиномиально недетерминированно вычислимая функция полиномиально вычислима.

Подсказка. Пусть заданы произвольные: полиномиальный алгоритм и его аргумент. Тогда вычисление, время которого ограничено полиномом $p(n)$, можно представить в виде:

- матрицы, состоящей из расположенных, скажем, сверху вниз, состояний вычисления (цепочек, возникающих в вычислении);
- в первой (верхней) строке матрицы записан аргумент, то есть исходное данное и догадка;
- каждый переход к следующей строке осуществляется в соответствии с функцией перехода алгоритма;
- самая нижняя строка, где есть не только пустые символы, содержит значение рассматриваемой вычислимой функции.

Размер матрицы является фиксированными полиномом от полинома $p(n)$.
Теперь существование вычисления нужно представить в виде выполнимости формулы логики высказываний со следующим смыслом:

- существует верхняя строка, в которой закодирована пара: исходное данное задачи (фиксированное для формулы) и догадка (существование которой мы выясняем);
- переход от тройки символов в любом месте таблице к следующей (вниз по вертикали) тройке осуществляется по функции перехода (фиксированной для формулы).

Формула будет использовать группы атомных высказываний, отвечающих:

- состоянию каждого символа таблицы;

- переходу от тройки символов к следующей.

Размер каждой такой одной группы ограничен константой, зависящей только от алгоритма, а не от аргумента.

Существование решения у нашей переборной задачи теперь эквивалентно выполнимости построенной формулы, где исходное данное фиксировано, а догадку надо подобрать. Размер формулы ограничен полиномом от исходного данного.

Идея **моделирования** любого вычисления **универсальным** также относится к числу больших идей, в следующем разделе мы увидим, как она используется в теории сложности объектов.