# Вычислимость и Ко

Калмыков Андрей

July, 2024

## Contents

# 1 Первый день. Машина Тьюринга

Прежде чем разбирать любые алгоритмы и их результаты, для начала нам нужно понять, на чём и как эти алгоритмы исполняются и что вообще такое алгоритм.

## Машина Тьюринга

Кто такой Тюринг и зачем ему машина?

Всё просто — это эдакий прапрапрадедушка компьютеров. Самая простая версия машины Тьюринга(МТ) состоит из *ленты* и *автомата* — набора состояний. Лента бесконечна в обе стороны и состоит из полей, поначалу пустых (они обозначаются спецсимволом $\lambda$, который не обязательно присутствует в алфавите). Также в МТ есть пишущая головка(ПГ), она постоянно указывает на некоторое место на ленте и знает, какой символ там сейчас находится. Символ этот принадлежит некоторому алфавиту, который также характеризует МТ.

Работа МТ заключается в выполнении действия, содержащегося в соответствующем состоянии и затем смене состояния на основании символа под ПГ (пару символ-состояние называют *конфигурацией МТ*)

Теперь о состояниях. Каждое состояние характеризуется символом, который видит ПГ, символом который надо записать в том месте, где ПГ находится, направлением, куда надо передвинуть ПГ (влево, вправо или оставить на месте) и состоянием в которое надо перейти. Также есть два особых состояния: стартовое, его обычно обозначают $q_0$ и конечное — $q!$, иногда ещё различаются два конечных состояния: $q_a$ (от слова accept) и $q_r$ (reject), машина не обязательно приходит в конечное состояние, она вполне может зациклиться и не прекращать работу.

*Попробуйте придумать пример подобной машины*

*Подумайте, сколько состояний может быть у машины (вопрос с подвохом)*

Если вы уже знакомы с программированием, то наверное уже недоумеваете: "Как МТ взаимодействует с внешним миром?" Даже в питоне изучение основ обычно начинается с изучения `input()`. Всё довольно просто — входом считается то, что записано на ленте изначально, то есть, если лента не пуста, а существует некоторео стартовое слово (последовательность символов). А выходом считается то, что находится на ленте после того, как машина остановилась (машина зациклилась — отдельный результат).

Заметим, что на ленте необязательно должно находится входное слово, его запись может быть "эмулирована" с помощью состояний МТ.

**Проблема останова**

Забегая вперёд стоит отметить, что МТ тоже можно как-то закодировать и соответственно подать другой машине на вход на вход другой (или той же самой МТ). Это ставит целый ряд новых вопросов и вот один из довольно интересных: "Можем ли мы придумать машину, которая останавливается и возвращает 1, если поданная машина не остановится на поданном входе, а иначен зацикливается?". Попробуйте дома подумать над этой проблемой, а завтра мы с вами её обсудим

*Неразрешимость проблемы останова:* Докажем это от противного. Допустим, Анализатор существует. Напишем алгоритм Диагонализатор, который принимает на вход число $N$, передает пару аргументов $(N, N)$ Анализатору и возвращает результат его работы. Другими словами, Диагонализатор останавливается в том и только том случае, если не останавливается алгоритм с номером $N$, получив на вход число $N$. Пусть $K$ — это порядковый номер Диагонализатора в множестве $S$. Запустим Диагонализатор, передав ему это число $K$. Диагонализатор остановится в том и только том случае, если алгоритм с номером $K$ (то есть, он сам) не останавливается, получив на вход число $K$ (какое мы ему и передали). Из этого противоречия следует, что наше предположение неверно: Анализатора не существует, что и требовалось доказать.

## Практика

На этом достаточно теории, пора поконтруировать эти самые МТ.

1. $A = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$. Пусть $P$ — непустое слово; значит, $P$ — это последовательность из десятичных цифр, т.е. запись неотрицательного целого числа в десятичной системе. Требуется получить на ленте запись числа, которое на 1 больше числа $P$

2. $A = \{a, b, c\}$. Перенести первый символ непустого слова $P$ в его конец

3. $A = \{a, b, c\}$. Если первый и последний символы (непустого) слова $P$ одинаковы, тогда это слово не менять, а иначе заменить его на пустое слово.

4. $A = \{a, b\}$. Удалить из слова $P$ его второй символ, если такой есть.

5. $A = \{a, b, c\}$. Удалить из слова $P$ первое вхождение символа $a$, если такое есть.

6. $A = \{a, b, c\}$. Удалить из $P$ все вхождения символа $a$.

7. $A = \{a, b, c\}$. Удвоить слово $P$, поставив между ним и его копией знак $=$.

## Домашнее задание

1. $A = a, b, c$. Заменить на $a$ каждый второй символ в слове $P$.
2. $A = \{a, b, c\}$. Определить, входит ли в слово $P$ символ $a$. Ответ: слово из одного символа $a$ (да, входит) или пустое слово (нет).
3. $A = \{0, 1, 2, 3\}$. Считая непустое слово $P$ записью числа в четверичной системе счисления, определить, является оно чётным числом или нет. Ответ: 1 (да) или 0 (нет)
4. $A = \{a, b, c\}$. Если $P$ – слово чётной длины $(0, 2, 4, \ldots)$, то выдать ответ $a$, иначе — пустое слово.
5. $A = \{0, 1, 2\}$. Считая непустое слово $P$ записью числа в троичной системе счисления, определить, является оно чётным числом или нет. Ответ: 1 (да) или 0.
6. $A = \{a, b, c\}$. Если слово $P$ имеет чётную длину, то оставить в нём только левую половину.
7. $A = \{a, b\}$. Перевернуть слово $P$ (например: $abb \to bba$)
8. $A = \{0, 1\}$. Считая непустое слово $P$ записью двоичного числа, получить это же число, но в четверичной системе.
9. $A = \{0, 1, 2, 3\}$. Считая непустое слово $P$ записью числа в четверичной системе счисления, получить запись этого числа в двоичной системе.
10. $A = \{|\}$. Считая слово $P$ записью числа в единичной системе, определить, является ли это число степенью 3 $(1, 3, 9, 27, \ldots)$. Ответ: пустое слово, если является, или слово из одной палочки иначе.
11. $A = \{|\}$. Считая слово $P$ записью числа $n$ в единичной системе, получить в этой же системе число $2^n$. $A = \{(, )\}$. Определить, сбалансировано ли слово $P$ по круглым скобкам. Ответ: Д (да) или Н (нет).
12. $A = \{a, b\}$. Если в $P$ символов $a$ больше, чем символов $b$, то выдать ответ $a$, если символов $a$ меньше символов $b$, то выдать ответ $b$, а иначе в качестве ответа выдать пустое слово.

Невостребованные задачи встретятся с вами на зачёте.

# 2 Второй день. Вычислимость

Сегодня мы наконец-то начинаем говорить про вычислимость, а также функции и множества, обладающие данным свойством

## 1. Вычислимые функции, разрешимые и перечислимые множества

### 1.1. Вычислимые функции

Функция $f$ с натуральными аргументами и значениями называется *вычислимой*, если существует алгоритм, её вычисляющий, то есть такой алгоритм $A$, что

- если $f(n)$ определено для некоторого натурального $n$, то алгоритм $A$ останавливается на входе $n$ и печатает $f(n)$;
- если $f(n)$ не определено, то алгоритм $A$ не останавливается на входе $n$.

Несколько замечаний по поводу этого определения:

1. Понятие вычислимости определяется здесь для частичных функций (областью определения которых является некоторое подмножество натурального ряда). Например, нигде не определённая функция вычислима (в качестве $A$ надо взять программу, которая всегда зацикливается).

2. Можно было бы изменить определение, сказав так: «если $f(n)$ не определено, то либо алгоритм $A$ не останавливается, либо останавливается, но ничего не печатает». На самом деле от этого ничего бы не изменилось (вместо того, чтобы останавливаться, ничего не напечатав, алгоритм может зацикливаться).

3. Входами и выходами алгоритмов могут быть не только натуральные числа, но и двоичные строки (слова в алфавите $\{0, 1\}$), пары натуральных чисел, конечные последовательности слов и вообще любые, как говорят, «конструктивные объекты». Поэтому аналогичным образом можно определить понятие, скажем, вычислимой функции с двумя натуральными аргументами, значениями которой являются рациональные числа.

Для функций, скажем, с действительными аргументами и значениями понятие вычислимости требует специального определения. Здесь ситуация сложнее, определения могут быть разными, и мы о вычислимости таких функций говорить не будем. Отметим только, что, например, синус (при разумном определении вычислимости) оказывается вычислимым, а функция $\mathrm{sign}(x)$, равная $-1$, 0 и 1 при $x < 0$, $x = 0$ и $x > 0$ соответственно — нет. Точно так же требует специального определения вычислимость функций, аргументами

которых являются бесконечные последовательности нулей и единиц и т. п.

4. Несколько десятилетий назад понятие алгоритма требовало специального разъяснения. Сейчас («компьютерная грамотность»?) такие объяснения всё равно никто читать не будет, поскольку и так ясно, что такое алгоритм. Но всё же надо соблюдать осторожность, чтобы не принять за алгоритм то, что им не является. Вот пример неверного рассуждения:

«Докажем», что всякая вычислимая функция $f$ с натуральными аргументами и значениями может быть продолжена до всюду определённой вычислимой функции $g\colon \mathbb{N} \to \mathbb{N}$. В самом деле, если $f$ вычисляется алгоритмом $A$, то следующий алгоритм $B$ вычисляет функцию $g$, продолжающую $f$: «если $A$ останавливается на $n$, то $B$ даёт тот же результат, что и $A$; если $A$ не останавливается на $n$, то $B$ даёт результат (скажем) 0». (В чём ошибка в этом рассуждении?)

## 1.2. Разрешимые множества

Множество натуральных чисел $X$ называется *разрешимым*, если существует алгоритм, который по любому натуральному $n$ определяет, принадлежит ли оно множеству $X$.

Другими словами, $X$ разрешимо, если его *характеристическая функция* $\chi(n) = (\textbf{if } n \in X \textbf{ then } 1 \textbf{ else } 0 \textbf{ fi})$ вычислима.

Очевидно, пересечение, объединение и разность разрешимых множеств разрешимы. Любое конечное множество разрешимо.

Аналогично определяют разрешимость множеств пар натуральных чисел, множеств рациональных чисел и т. п.

**1.** Докажите, что множество всех рациональных чисел, меньших числа $e$ (основания натуральных логарифмов), разрешимо.

**2.** Докажите, что непустое множество натуральных чисел разрешимо тогда и только тогда, когда оно есть множество значений всюду определённой неубывающей вычислимой функции с натуральными аргументами и значениями.

Отметим тонкий момент: можно доказать разрешимость множества неконструктивно, не предъявляя алгоритма. Вот традиционный пример: множество тех $n$, для которых в числе $\pi$ есть не менее $n$ девяток подряд, разрешимо. В самом деле, это множество содержит либо все натуральные числа, либо все натуральные числа вплоть до некоторого. В обоих случаях оно разрешимо. Тем не менее мы так и не предъявили алгоритма, который по $n$ узнавал бы, есть ли в $\pi$ не менее $n$ девяток подряд.

# 3 Третий день

# 4 Четвёртый день

# 5 Пятый день

# 6 Зачёт