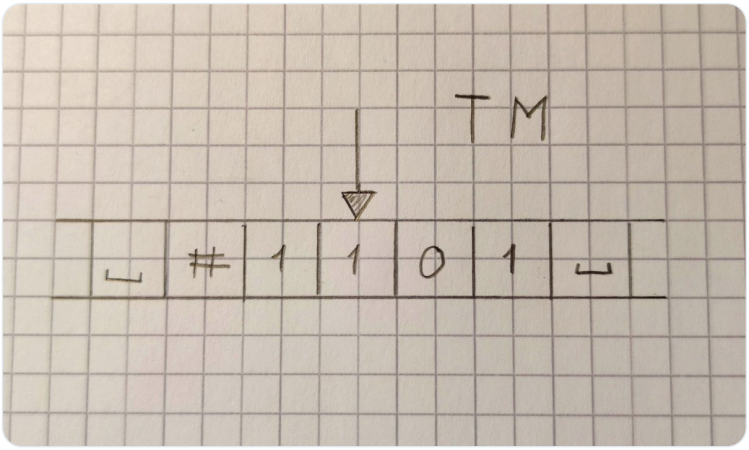
Валерий Жила @ValeriiZhyla

Oct 18, 2021 • 61 tweets • ValeriiZhyla/status/1450015818385854465

Я расскажу очень простым языком о том, как устроены Тьюринг Машины, и зачем они вообще нужны!



Уверен, что вы уже где-то слышали про Алана Тьюринга, его заслуги и личную жизнь, и видели упоминания каких-то странных машин Тьюринга, но никогда особо не вникали в тему

Когда я впервые с этим столкнулся, и начал копать интернет, у меня разболелась голова. Какие-то ленты, какие-то устройства, очень много абстракций каких-то, всё странно и ничего не понятно. Это реальная машина? Её кто-то построил? Зачем она вообще?

Сейчас мне смешно об этом вспоминать, потому что эта самая машина – концепция до смешного простая. Простая, как пять копеек. Чуть-чуть сложнее конечных автоматов (потому что во всю их эксплуатирует).

Если кто-то в твиттере ещё не совсем хорошо понимает, как работают конечные автоматы – вот мой свежий тред на эту тему:

Итак, в бой.

Тьюринг Машины, они же Машины Тьюринга – я буду называть их далее просто ТМ

Обычная ТМ состоит из трех частей, из которых первые две сугубо технические, а в третьей происходит магия.
Эти три части, по порядку: головка чтения и записи, бесконечная лента с ячейками, и программа машины

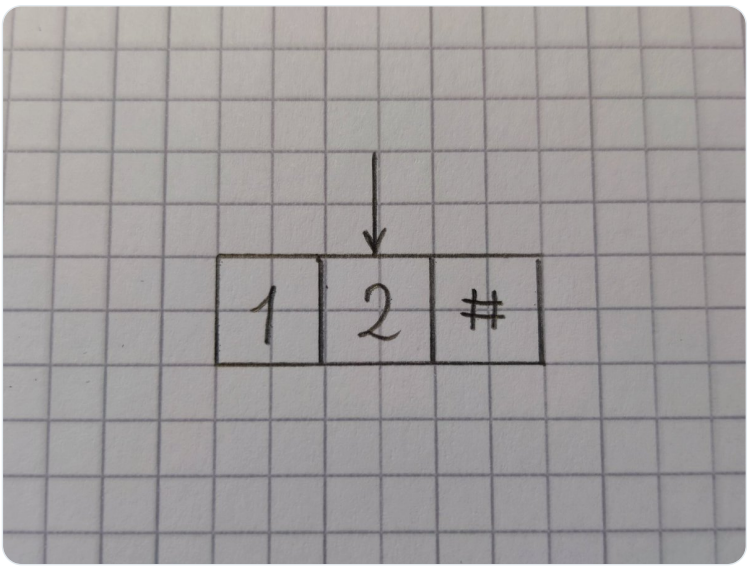
Очень важно сразу понять, что ТМ – это очередная абстракция из мира Computer Science. В физическом виде её не существует, но в конце треда поговорим и об этом. Это очень-очень полезная штука, потому что возможности обычных конечных автоматов довольно скудны.

Итак – запчасти!

Головка чтения и записи – мне не очень нравится это название. А сокращать как ГВЗ мне не хочется, некрасиво. Никто не будет против, если я неформально буду называть эту запчасть указателем?

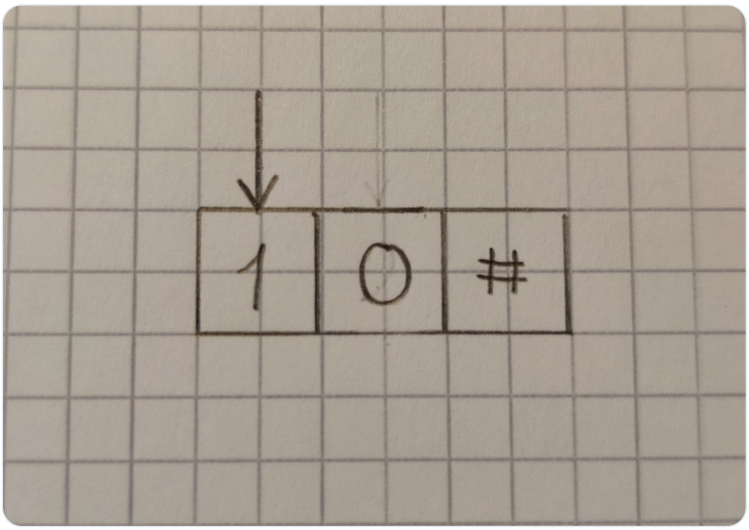
Единогласно, указатель так указатель. Указатель обозначается вот такой вот стрелкой, и всегда указывает на одну ячейку. Этот элемент может ровно три вещи: он читает содержимое ячейки, он записывает что-то в ячейку, и он перемещается

Вот маленький пример: указатель "стоит" на ячейке, в которой содержится двойка. Сейчас он может прочитать её содержимое, перезаписать ячейку или поменять своё положение



Указатель может двигаться только тремя способами – на одну ячейку влево, на одну ячейку вправо или остаться стоять на месте.

Давайте запишем в ячейку ноль, и после этого переместим наш указатель на одну ячейку влево. Выглядеть результат будет так:



Все действия машины строятся следующим образом: читаем ячейку, что-то пишем в ячейку (в зависимости от программы и прочитанного), перемещаемся.

Действие, которое мы сделали выше, можно описать как (2, 0, L)
Читаем 2, пишем ноль, едем налево

Движения обозначают часто буквами L, N, R – Left, None, Right – едем налево, стоим на месте, едем направо

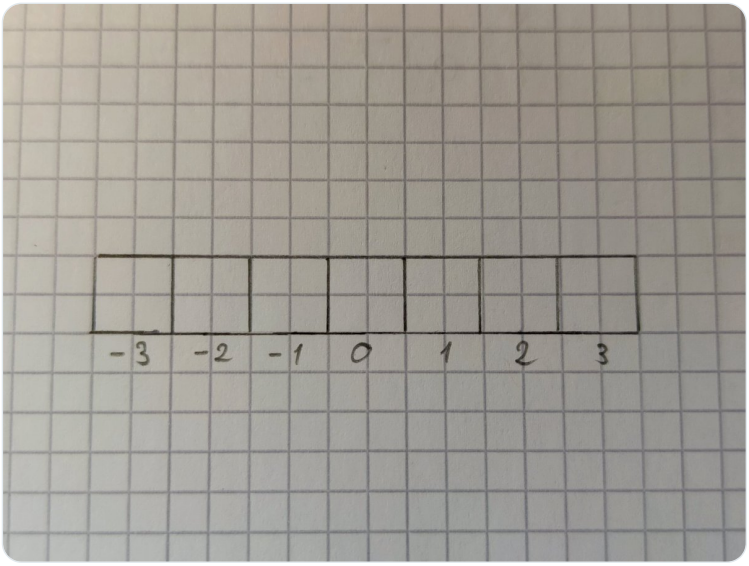
С указателем, он же головка чтения записи, мы разобрались. Но, спросите вы, что это за ячейки, что за программа, откуда ты всё это взял?!

Всё по порядку!

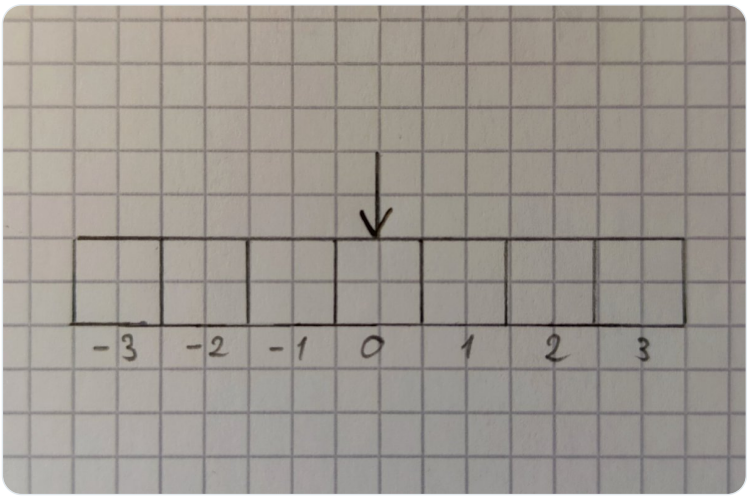
Следующий элемент машины, который и делает ТМ настолько мощной – её бесконечная лента.

Бесконечная лента? Как это вообще понимать?

В примере выше была лента из трех ячеек. Этого точно маловато.
Возьмём ленту чуть побольше: из семи ячеек! Я пронумеровал их целыми числами от -3 до 3

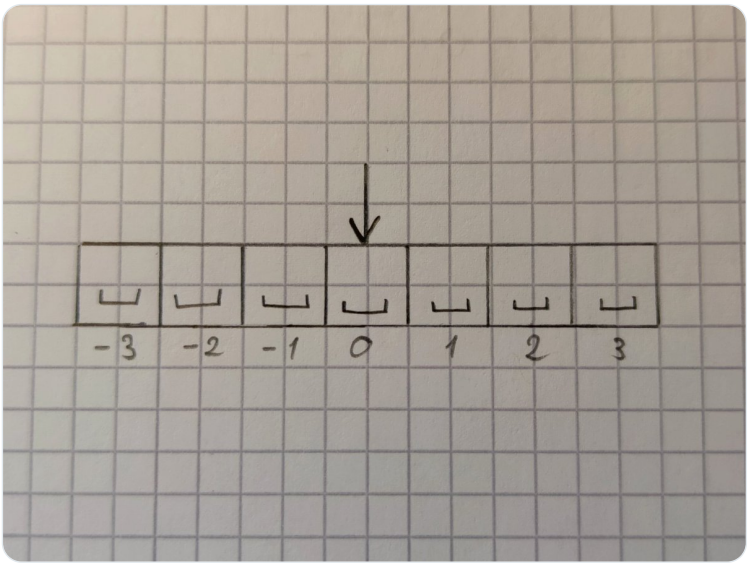


Примем за правило, что указатель находится начале работы машины всегда над ячейкой с номером 0



Важно сказать, что эти номера тут только для наглядности, в настоящей ТМ их нет, и она не может "прыгнуть" на ячейку с конкретным номером. Никакого Random Access у них нет, в общем.

Ячейки по умолчанию заполнена "пустыми символами". Это делают для того, чтоб можно было проверять, пуста ли ячейка, как бы читая символ из неё. Пустые символы обозначают либо просто пустыми ячейками, либо явно символом space:

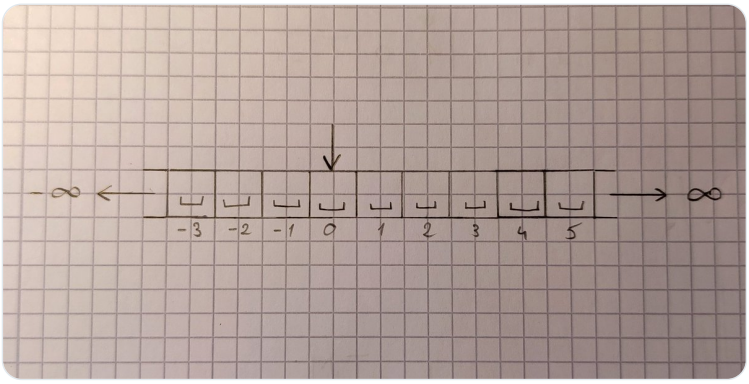


Хорошо, лента с 7 пустыми ячейками у нас имеется, но ты же обещал бесконечную ленту!

Да, всё верно, и я постараюсь сдержать свое слово

Лента ТМ похожа на нашу ленту, но состоит не из семи ячеек, а бесконечна направо и налево.

Так как никакой нумерации в ТМ нет, нам достаточно того, что мы начинаем всегда в середине ленты, и можем бесконечно долго двигаться в обе стороны, читать с ленты и записывать на ленту.



В основном, ТМ бывают двух видов:

1) ТМ, которые начинают с входными данными на ленте, и должны как-то преобразовать данные – ТМ, которые считают. Умножают, делят, сортируют массивы..

Эти ТМ "считают", в общем смысле этого слова, оставляют результат на ленте, и останавливаются

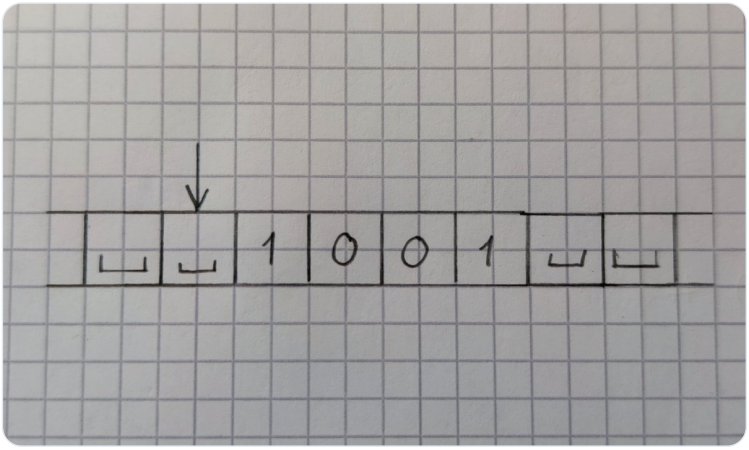
2) ТМ, которые начинают с входными данными на ленте, и используют их для того, чтоб дать ответ ДА/НЕТ.
Истинно ли выражение? Чётно ли число? Отсортирован ли массив? Связан ли граф?

Эти ТМ "решают", и останавливаются в состоянии TRUE или FALSE

ТМ второго типа намного важнее и интереснее машин первого типа для Computer Science. Поэтому дальше я буду говорить только о них. На самом деле, второй тип даже сложнее первого

На ленте могут стоять только заранее определённые символы, и они не могут добавляться по ходу работы машины. Для большинства задач с входными данными, закодированными двоичным кодом, вполне удобно пользоваться тремя символами 1,0 и # для разграничения. Плюс "пустой" символ

Вводные данные машины, например, число, для которого нужно проверить, является ли оно простым, стоят перед запуском машины в выбранной кодировке на ленте, справа от указателя. Например, вот так бы выглядело число 9 (=1001 в двоичной системе) в качестве вводных данных:



Надеюсь, к ленте вопросов больше не осталось. Если что-то не ясно – спрашивайте, постараюсь ответить)

У нас впереди самое сочное – последняя часть машины, её программа!

Программа машины отвечает за логику движения указателя, за "состояния" и "переходы" между состояниями, которые для этого необходимы. Именно в программе содержится то, что делает каждую ТМ уникальной.

До этого мы разбирали общие вещи для всех ТМ – у них всех есть бесконечная лента, у них всех есть указатель, а вот программа – это то, что отличает машину, проверяющую числа на простоту от машины, считающую связность графа.

Состояния, переходы между состояниями – такое чувство, словно это где-то уже было, не находите?

О Господи, конечные автоматы, точно. Ты хочешь нам сказать, что ТМ это модификация конечных автоматов, которые могут работать с внешним хранилищем информации (лентой ТМ)?!

Что я могу сказать? Этим ТМ и является!

И программу ТМ удобнее всего представлять именно конечным автоматом, который перемещается между состояниями, пишет на ленту, читает с ленты, и двигает указатель

Конкретную программу в студию!

Для наглядности возьмём программу, которая отзеркаливает последовательность 0 и 1, стоящую на ленте. То есть делает настоящий String.reverse! Например, 01011 будет трансформировано в 11010

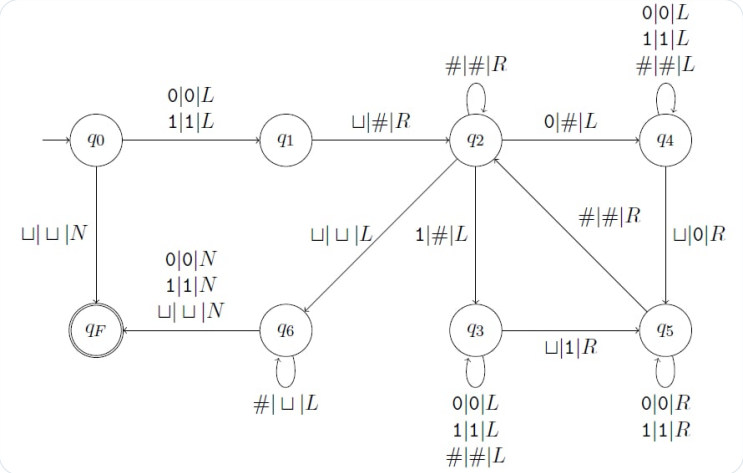
Тут я сделаю два замечания: программы ТМ зачастую большие и сложные, делают кучу странной мишуры, постоянно ездят из конца слова в другой конец. Это вытекает из максимальной простоты инструментов ТМ.

И тем не менее, программа ТМ существует для каждой алгоритмическт решаемой задачи! Что это значит?

Всё, что априори может быть вычислено, может быть вычислено на ТМ. Скорее всего, за монструозное время, никаких тебе O(nlogn), скорее за O(n^42) или даже O(n^n).

Вот так будет выглядеть программа ТМ, которая будет переворачивать строку. Я не хочу её выстраивать по шагам, или подробно разбирать – даже такая простая вещь у меня несколько лет назад заняла часа два на то, чтоб её придумать.

Смысл этого примера – один раз увидеть программу ТМ



Как именно она работает?
Указатель едет к слову, пишет перед ним # – разграничитель. Потом "отщипывает" самую левую букву от слова, заменяя её на # и запоминая букву. Дальше катится налево, и пишет эту букву за #.

И так потихонечку, по одной буковке, катаясь туда сюда, ТМ отзеркалит изначальное слово. Когда слово закончится, останется только стрететь все # с ленты и закончить работу

Запутанно? Да, есть немного. Вот вам пример работы этой программы для слова 10, состоящего всего из двух букв:

$$(q_0)10 \to (q_1)\sqcup 10 \to \#(q_2)10 \to (q_3)\#\#0 \to (q_3)\sqcup \#\#0 \to 1(q_5)\#\#0 \to 1\#(q_2)\#0 \to 1\#\#(q_2)0 \to 1\#(q_4)\#\# \to 1(q_4)\#\#\# \to (q_4)1\#\#\# \to (q_4)\sqcup 1\#\#\# \to 0(q_5)1\#\#\# \to 01(q_5)\#\#\# \to 01\#(q_2)\#\# \to 01\#\#(q_2)\# \to 01\#\#\#(q_2)\sqcup \to 01\#\#(q_6)\# \to 01\#(q_6)\# \to 01(q_6)\# \to 0(q_6)1 \to 0(q_F)1$$

Поздравляю вас! Запутанная программа закончилась!

Теперь начинается лирика)

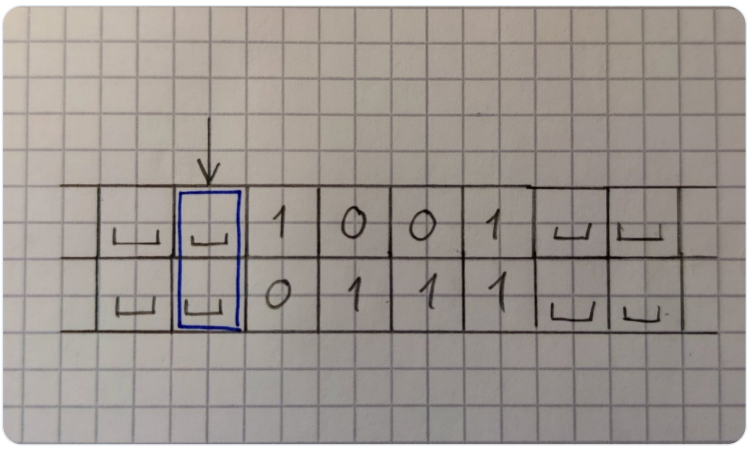
Во-первых, простите, если напугал этим примером. Не хотелось рассказывать про ТМ, не показав конкретной ТМ с конкретными данными. Мне самому плохеет, когда я смотрю на их программы.

Во-вторых, хорошие новости для Computer Scien'тистов – обычно создавать автоматы для доказательств не нужно. Это слишком сложно для человека – у ТМ для поиска корней квадратного уравнения может быть за сотню состояний в программе.

Так что обычно описание ТМ делается в текстовом виде, ближе к обычной программе: едем к слову, вырезаем букву и пишем её левее самого левого элемента на ленте, едем назад, продолжаем, пока правое слово не кончится.

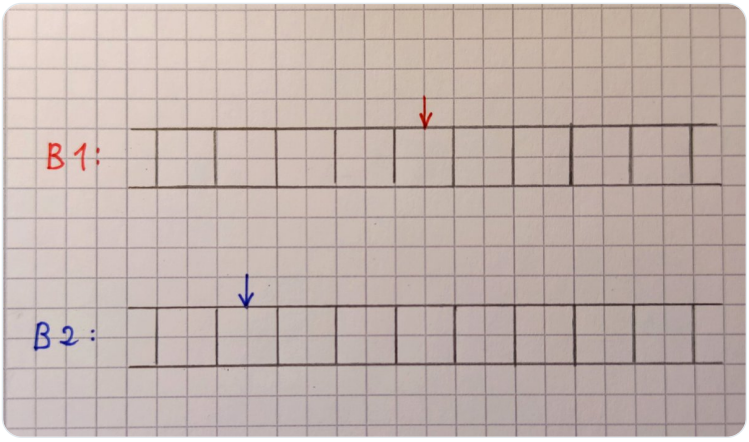
При правильной структуре таких текстовых описаний можно приноровиться быстро считать их алгоритмическую сложность по времени и памяти.

В-третьих – машина машине рознь. Есть модификации с лентами шириной не в 1, а в несколько ячеек:



В таких ТМ считываться будет сразу целое окно ячеек (вектор)

Ещё есть модификация ТМ, где используется несколько лент и несколько указателей в рамках одной машины:



И ещё много, много видов ТМ: с настраиваемой дальностью прыжка указателя, например

Прелесть этих модификаций в том, что их всех можно симулировать на обычной ТМ, с одной лентой и с одним указателем – значит, и решают они ровно те же задачи. А значит, что обычная ТМ может не меньше, чем ТМ с тысячей лент и указателей.

А это значит, что можно решать задачи на модификациях ТМ, что в разы удобнее, и писать, мол, доказано, что эту ТМ можно симулировать на обычной ТМ. Вуаля!

В-четвертых: ТМ не всегда останавливаются, и точно так же, как обычные программы, могут попасть в бесконечный цикл. Это служит почвой для десятков парадоксов, вроде Проблемы Остановки, которые широко используются в теории вычислимости

В-пятых: если в основе программы ТМ лежит детерминистичный конечный автомат (один переход в другое состояние для каждого символа из каждого состояния), то и сама ТМ будет являться детерминистичной. Сокращённо – ДТМ

Если автомат недетерминистичный, то, соответственно, такой будет и ТМ. Сокращённо – НДТМ

В одном из следующих тредов я буду говорить о недетерминизме. Слышали про классы P и NP? Это связано как раз с тем, использует программа ДТМ или НДТМ.

Вот, в целом, и всё, что человеку нужно знать про Машины Тьюринга. Надеюсь, что у меня получилось удержать градус сложности в рамках разумного!

@threadreaderapp unroll

• • •