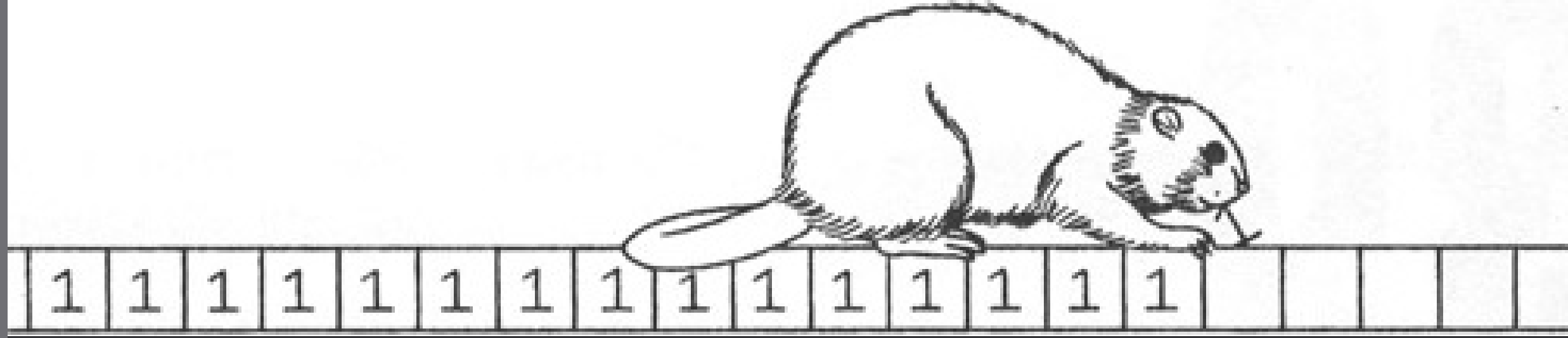
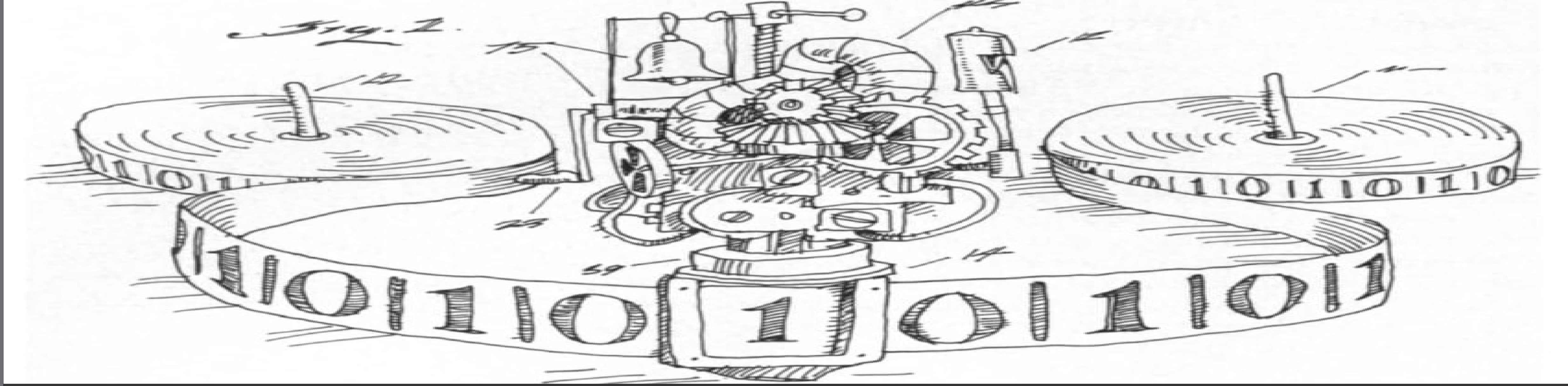


The Busy Beaver Problem

S. Mohsen Sadeghi

Mehrab Kalantari



Turing Machine

The way Turing described this machine goes like this:

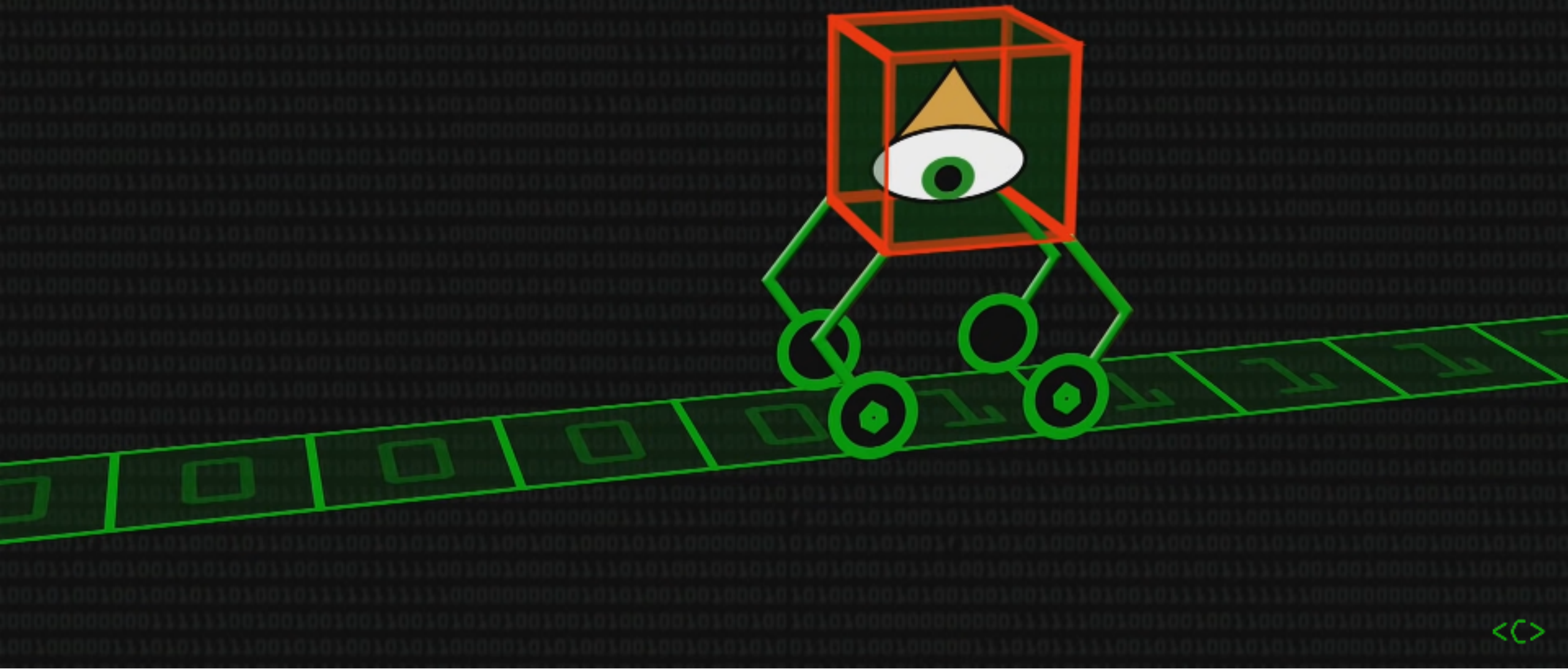
You have a way of writing down information in a coded form.

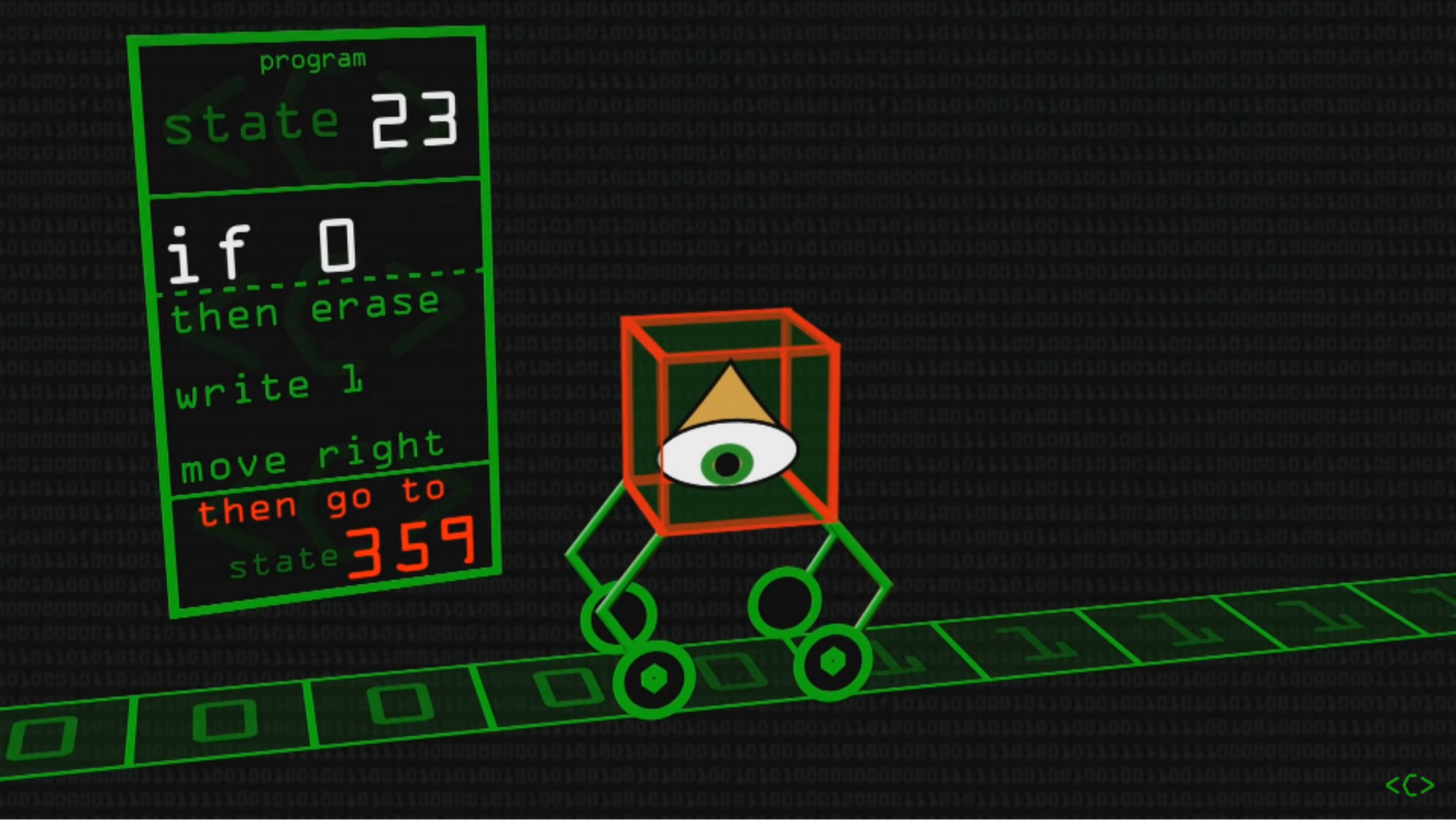
What Is A Turing Machine?

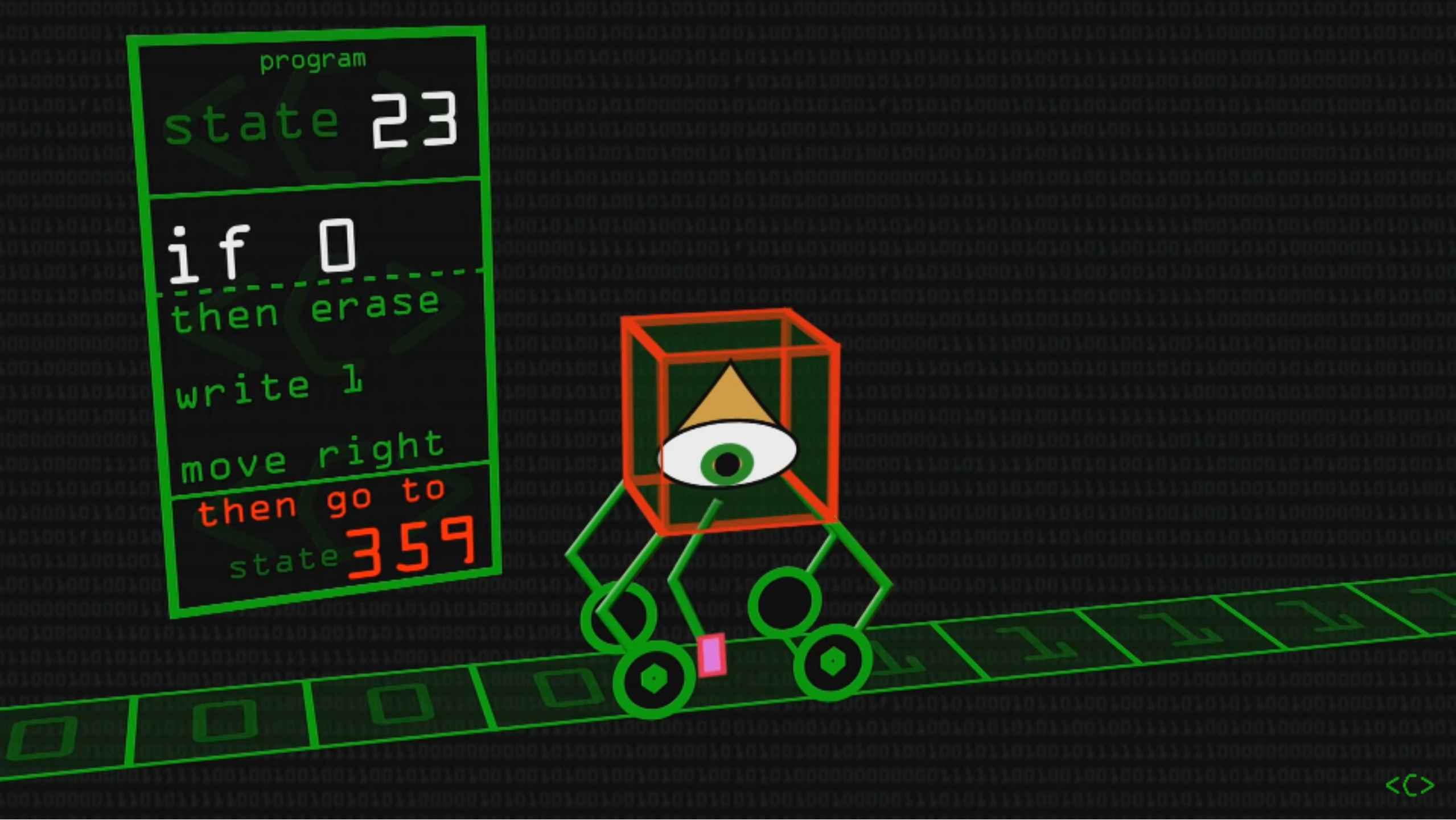
A tape which is a long as it needs to be.

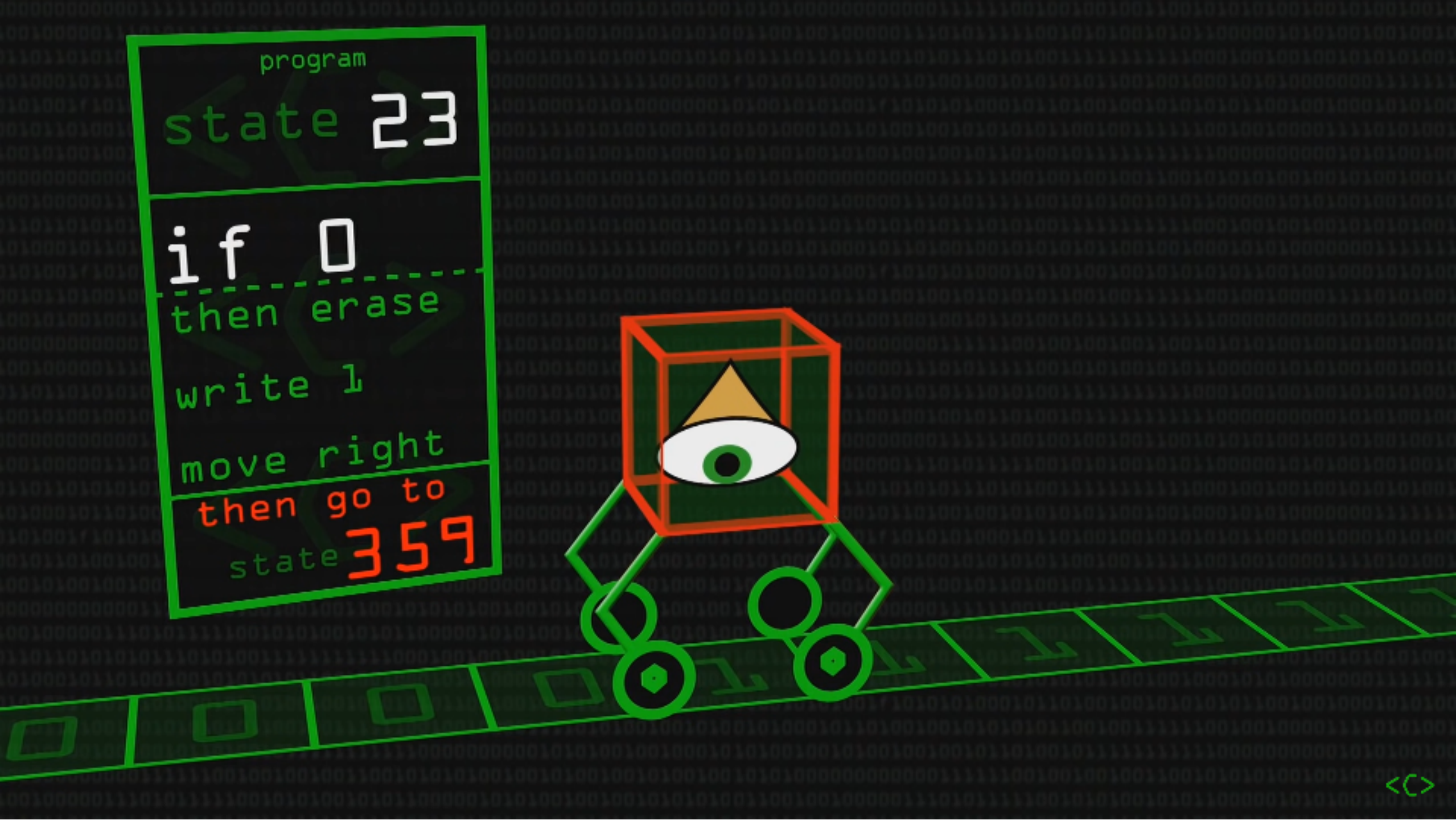
Its divided up into squares.

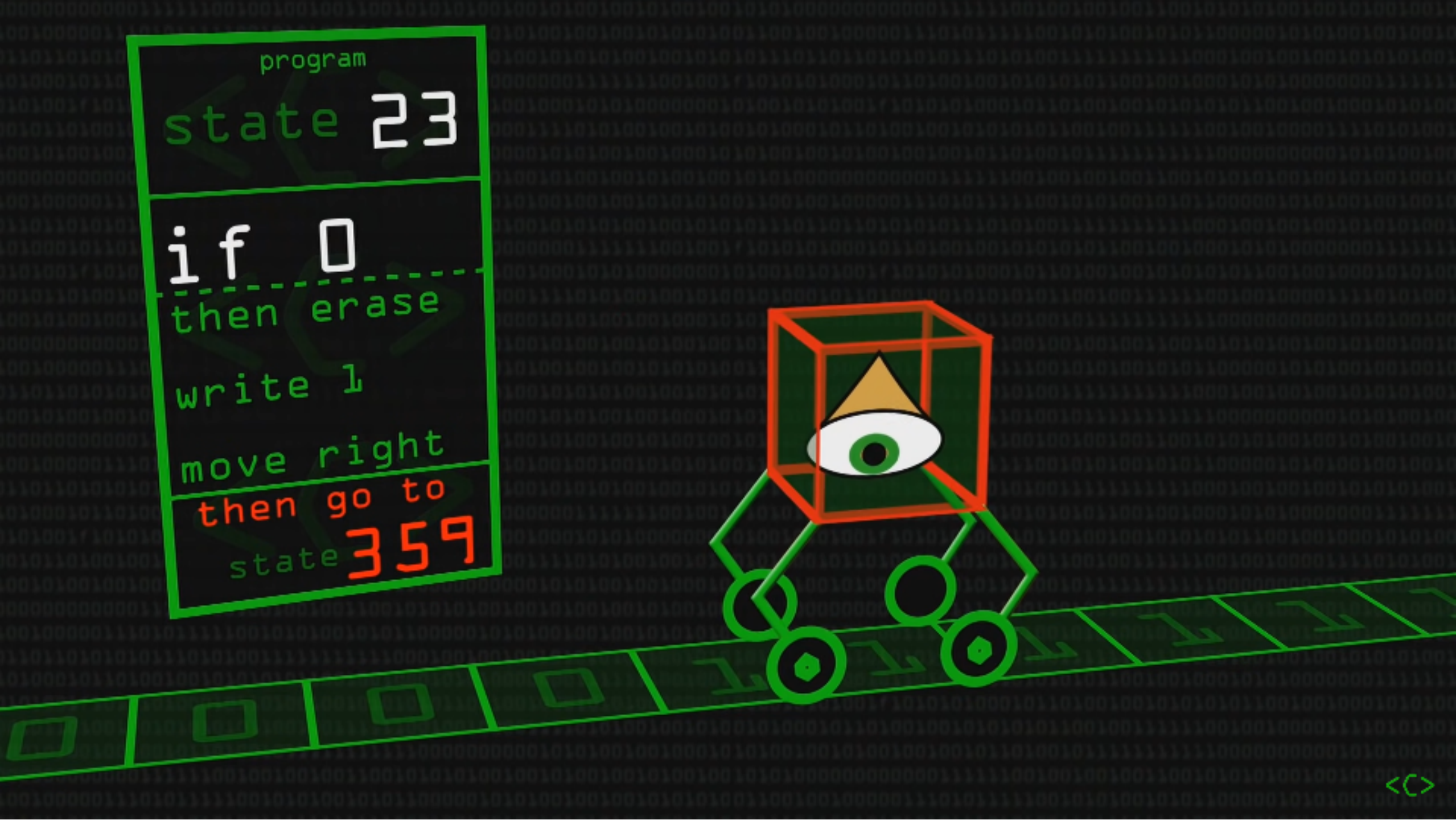
And the machine looks at the tape, one square at a time.




program
state 23
if 0
then erase
write 1
move right
then go to
state 359


program
state 23
if 0
then erase
write 1
move right
then go to
state 359


program
state 23
if 0
then erase
write 1
move right
then go to
state 359


program
state 23
if 0
then erase
write 1
move right
then go to
state 359

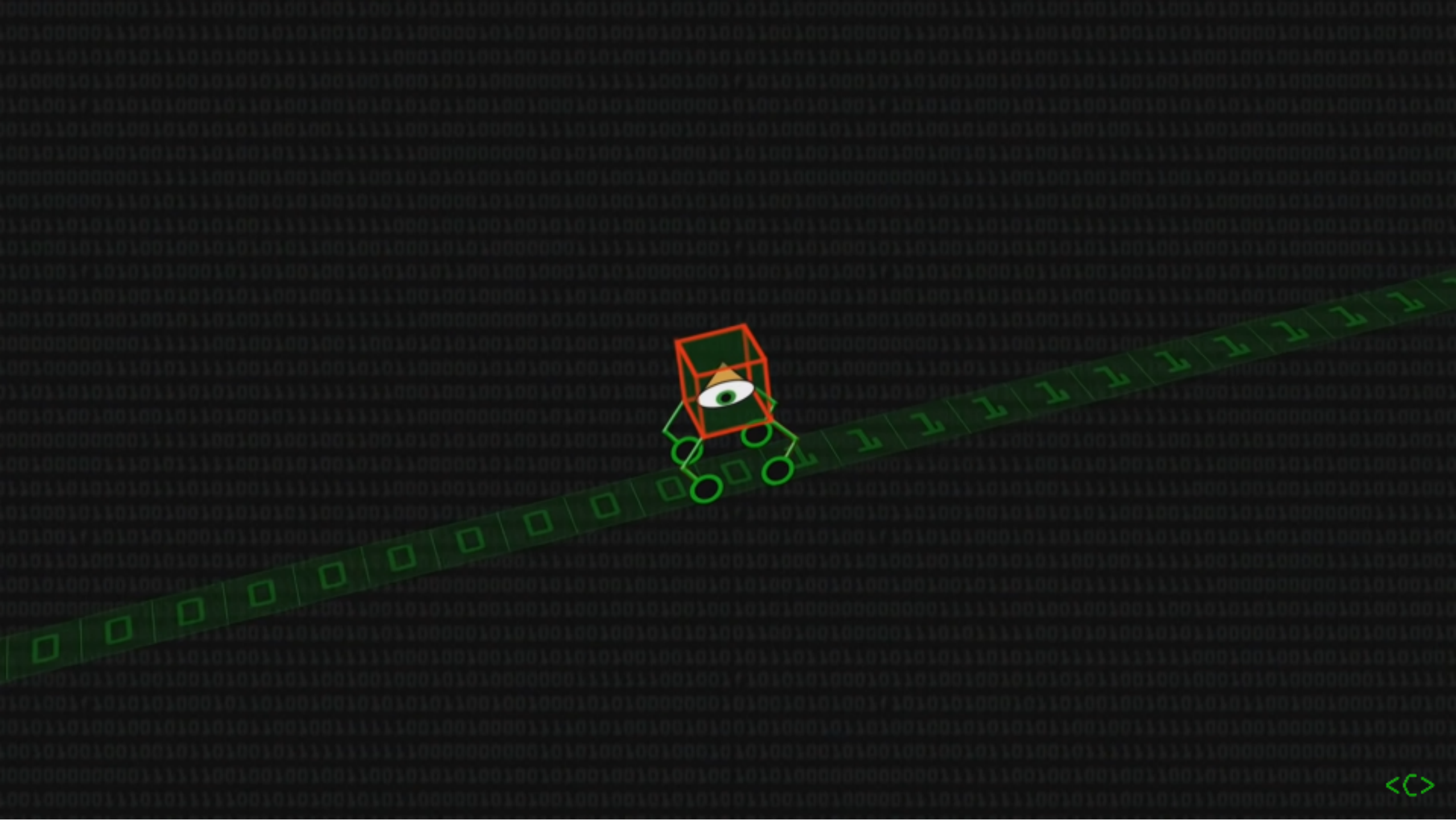
What Turing Machine Does

- It starts of with a certain pattern of ones and zeroes.
- It follows the rules one square at a time and changes that string of ones and zeroes into a different string of ones and zeroes.
- After that

  machine moves into a halting state (accepted)

  and what's left on the tape is the answer to out problem.

  (coded up with ones and zeroes)



Turing Machine

- $M = (Q, \sum, \Gamma, c, q_0, b, F)$
- Q : states
- $\sum$: input symbols
- Γ or S : tape symbols
- c : transition function
- $c : Q \times \Gamma \rightarrow Q \times \Gamma \times \{ R, L \}$
- q_0 : initial state
- b : blank symbol
- F or H : final states or halting states

Computable Function

A computable function is a function:

$$\mathbf{f: \sum^* \Rightarrow \sum^*}$$

that can be computed by a Turing Machine.

The input X is on the tape.

The TM runs and **ALWAYS HALTS.**

The result f(X) is left on the tape.

Church-Turing Thesis

It states that a function on the natural numbers can be calculated by an effective method if and only if it is computable by a Turing machine.

Note that this isn't a formal statement, so we cant prove it. But this thesis has near-universal acceptance.

Halting Problem

- Given a computer program and an input, will the program terminate or will it run forever?
- The question is simply whether the given program will ever halt on a particular input.
- It can be proven that this is undecidable

Ackerman Function

- A program that will eventually halt but the answer to it can never be known

$$A(m,n) = \begin{cases} n + 1 & \text{if } m=0 \\ A(m-1,1) & \text{if } m > 0 \text{ and } n = 0 \\ A(m-1,A(m,n-1)) & \text{if } m > 0 \text{ and } n > 0 \end{cases}$$

where m and n are non-negative integers

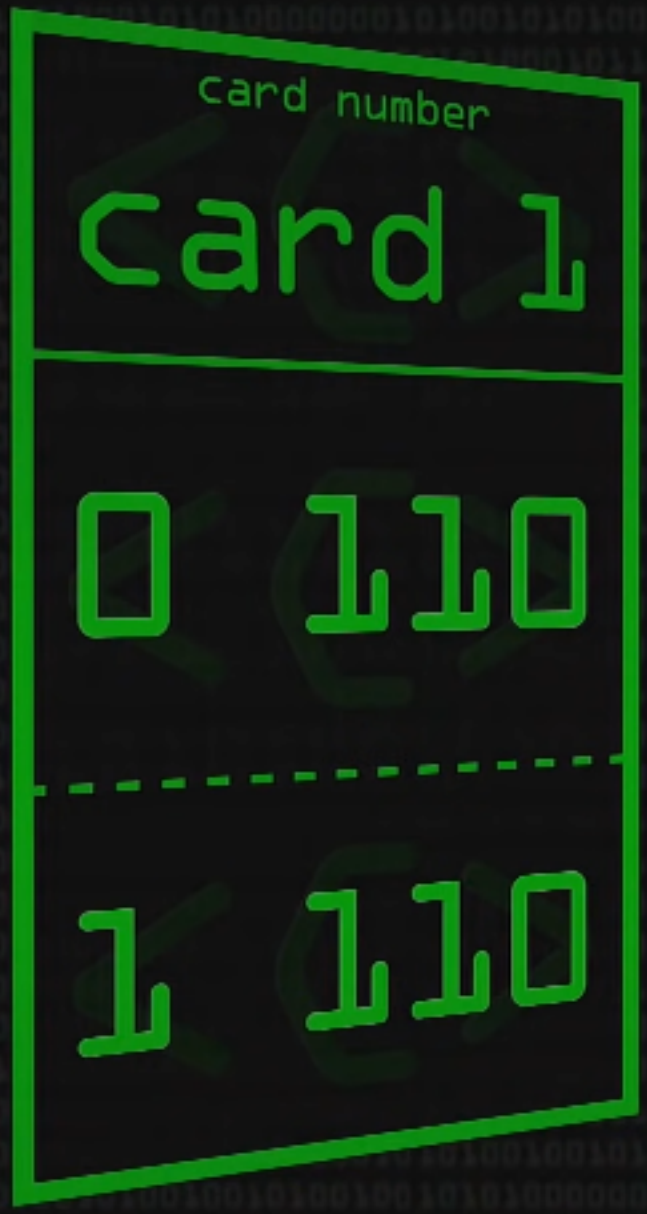
What Is Busy Beaver Problem?

- How long can a simple computer program possibly run before it terminates?
- given an n-state Turing Machine with a two symbol alphabet {0, 1},
- what is the maximum number of 1's that the machine can print on an initially blank tape (filled with 0's) before halting?

The Busy Beaver Function

- $B(1) = 1$
- $B(2) = 4$
- $B(3) = 6$
- $B(4) = 13$
- $B(5) = 4098$ (the exact result has not yet been found)
- $B(6) = 4.6 \times 10^{1439}$ (the exact result shall never be known)

<busy beaver turing machine>

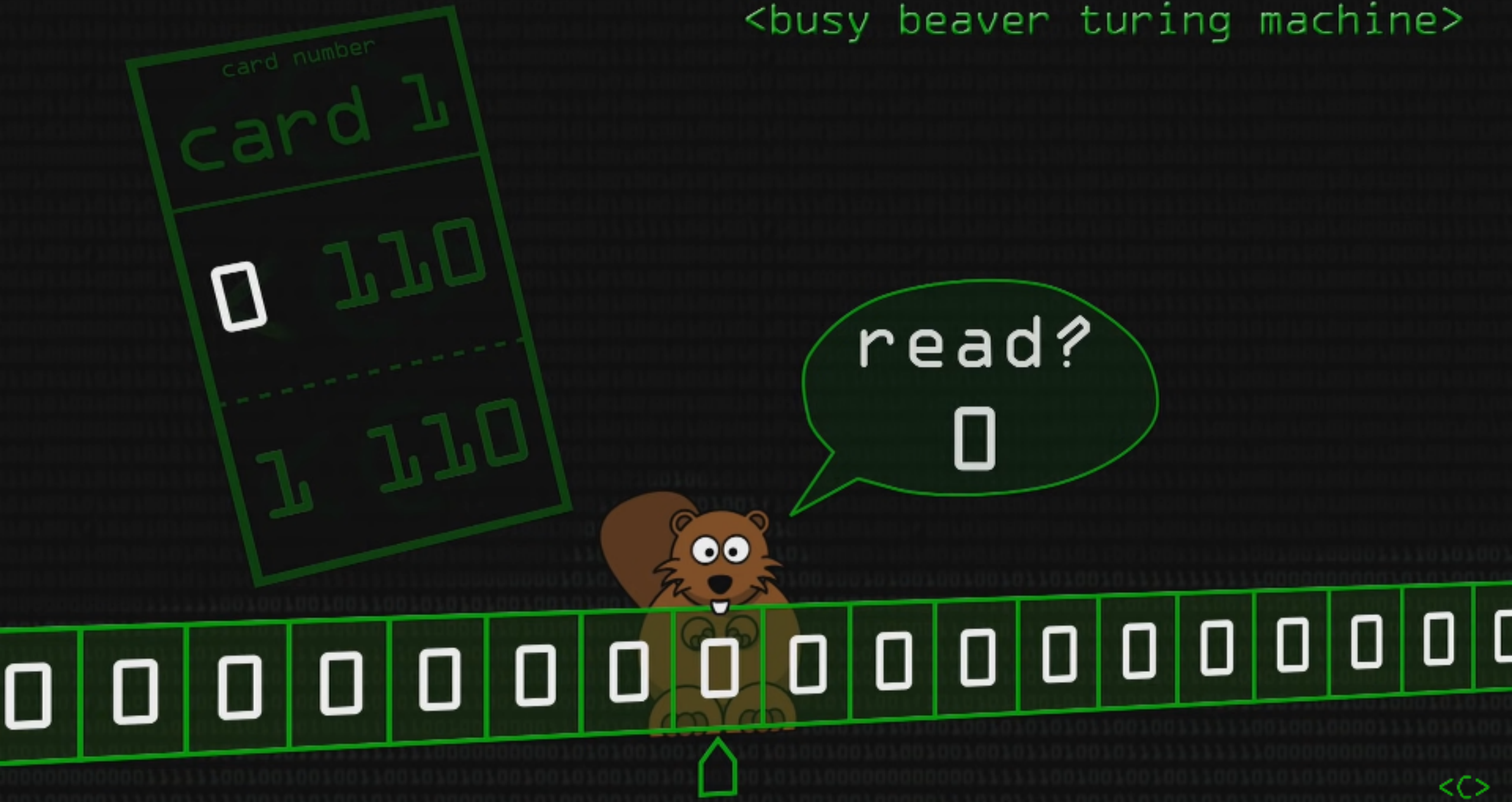


<C>

<busy beaver turing machine>

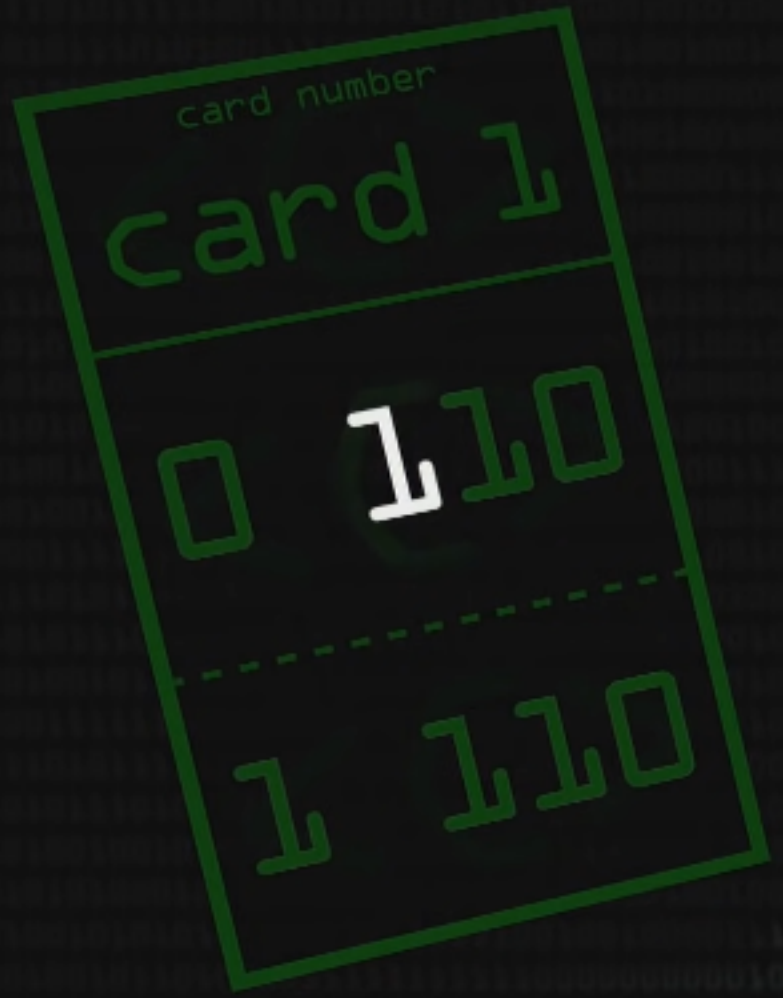
<C>

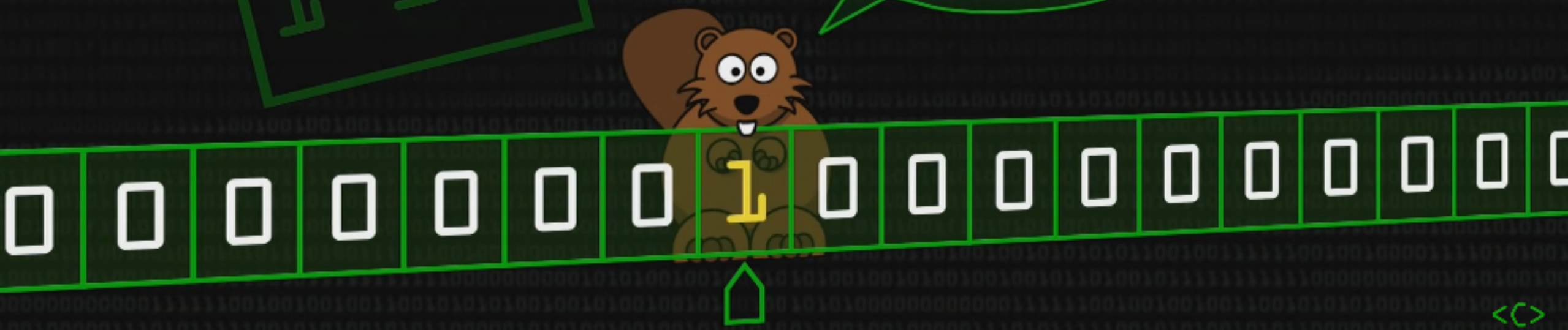
<busy beaver turing machine>



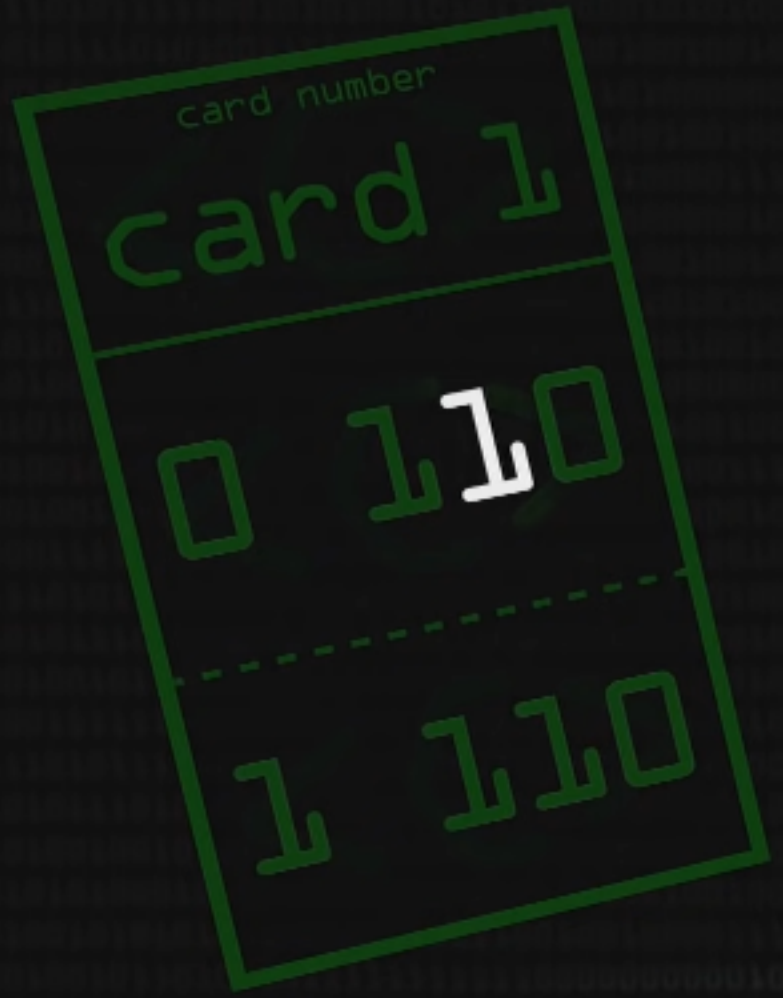
<C>

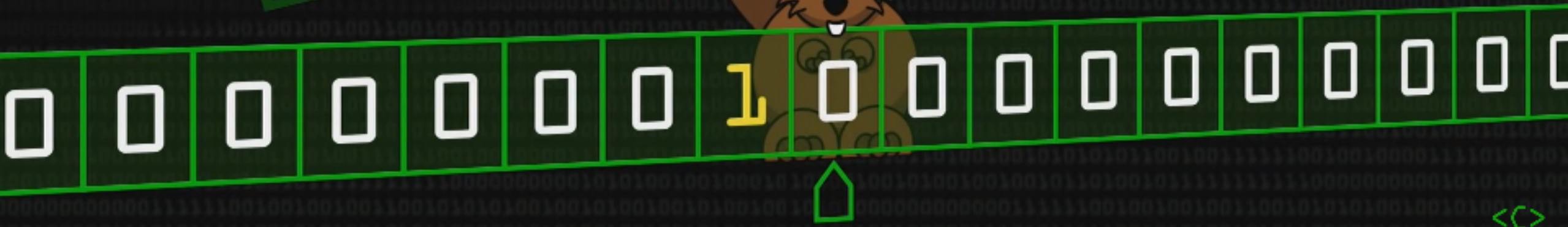
<busy beaver turing machine>



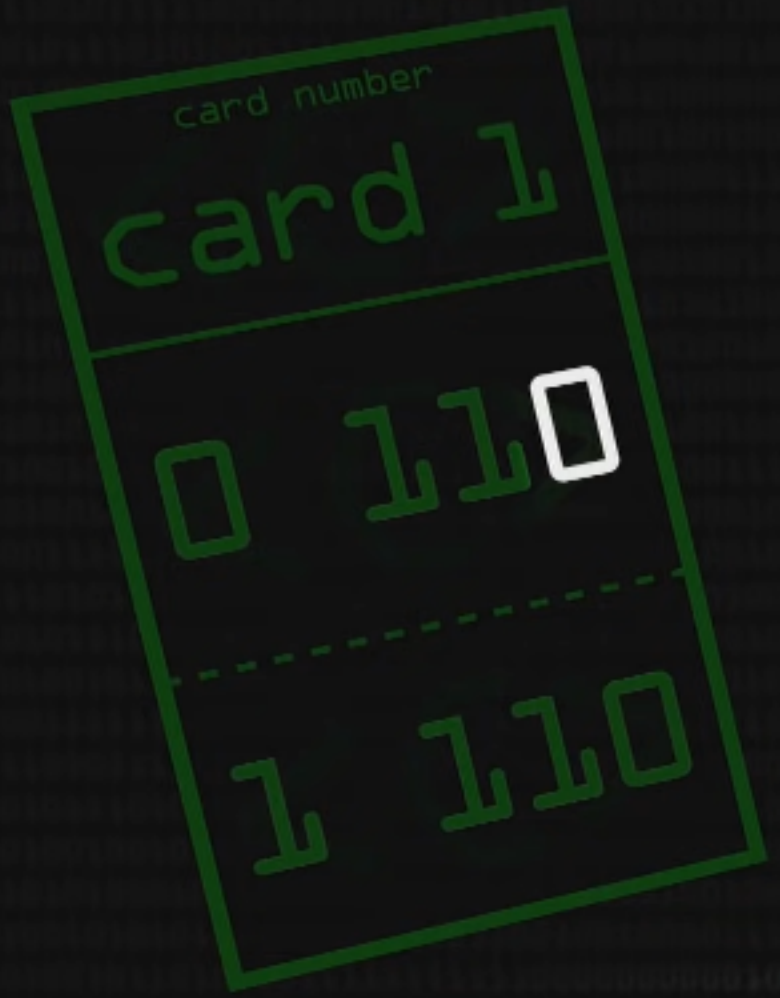


<busy beaver turing machine>





<busy beaver turing machine>



<C>

Non-Computable Function

The list of all computable functions is written, and then a function is defined such that it is distinct from each function in the list. Then this function is noncomputable.

Turing proved that there's no reliable, repeatable method for distinguishing machines that halt from machines that simply run forever — a fact known as the halting problem.

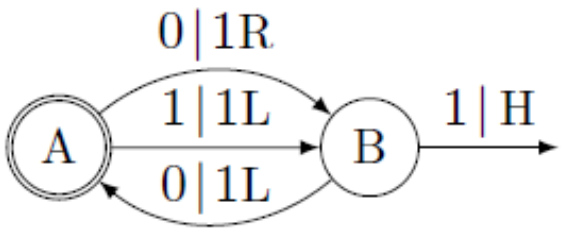
In 1962, Rado succeeded in providing a natural definition for noncomputable functions on the integers. He defined a *Busy Beaver* game, leading to two functions Σ and S which are still the best examples of noncomputable functions that one can give nowadays. The values (n) and S(n) are defined by considering the finite set of carefully defined Turing machines with two symbols and n states, and picking among these machines those with some maximal behavior.

Functions Σ and S

- In order to define functions Σ and S, Rado (1962) considers Turing machines with n stated and two symbols 0 and 1. There are exactly $\left(2 \times 2 \times (n+1)\right)^{2n}$ Turing machines that fit the given description. Some of these machines never stop (never reach the halting state). The other ones that eventually do stop, are called busy beavers, and they are competing in two competitions, for the maximum number of steps and for the maximum number of non-blank symbols left on the tape.

$$S(n) = \max\{s(m): M \text{ is a busy beaver with n states}\}$$
$$\Sigma(n) = \max\{\sigma(m): M \text{ is a busy beaver with n states}\}$$

note that this could be easily extended to Turin machines with more than 2 states but for the sake of simplicity we will only analyze the 2 symbol machines.



initial state	···	0	0	0	0A	0	0	···
step 1	···	0	0	0	1	0B	0	···
step 2	···	0	0	0	1A	1	0	···
step 3	···	0	0	0B	1	1	0	···
step 4	···	0	0A	1	1	1	0	···
step 5	···	0	1	1B	1	1	0	···
step 6 (halt)	···	0	1	1H	1	1	0	···

$n = 2$	A	B
0	1RB	1LA
1	1LB	H

$n = 3$	A	B	C
0	1RB	1LB	1LC
1	H	0RC	1LA

$n = 4$	A	B	C	D
0	1RB	1LA	H	1RD
1	1LB	0LC	1LD	0RA

$n = 5$	A	B	C	D	E
0	1RB	1RC	1RD	1LA	H
1	1LC	1RB	0LE	1LD	0LA

$n = 6$	A	B	C	D	E	F
0	1RB	1RC	1LD	1RE	1LA	H
1	1LE	1RF	0RB	0LC	0RD	1RC

$n = 7$	A	B	C	D	E	F	G
0	1RB	1RC	1LD	1LF	H	1RG	1LB
1	N/A	0LG	1RB	1LE	1LF	0LD	0RF

Threshold Of Unknowability

- Mathematicians are using the game as a yardstick for gauging the difficulty of important open problems in mathematics — or for figuring out what is mathematically knowable at all.
- The Goldbach conjecture (very even integer greater than 2 is the sum of two primes)
- Code Golf Addict published code for a 27-rule Turing machine that halts if — and only if — the Goldbach conjecture is false.
- Whether near or far, such thresholds of unknowability definitely exist. “This is the vision of the world that we have had since Gödel,” said Aaronson. “The busy beaver function is another way of making it concrete.”

References

- https://www.youtube.com/watch?v=CE8UhcyJS0I
- https://hal.archives-ouvertes.fr/hal-00396880v6/document
- http://logic.amu.edu.pl/images/6/6c/Bobr.pdf
- https://www.scottaaronson.com/papers/bb.pdf
- https://web.stanford.edu/class/cs54n/handouts/15-UncomputableFunctions.pdf
- https://warwick.ac.uk/fac/soc/philosophy/people/miller/dm_bbeaver.pdf
- https://www.quantamagazine.org/the-busy-beaver-game-illuminates-the-fundamental-limits-of-math-20201210/
- https://gist.github.com/anonymous/a64213f391339236c2fe31f8749a0df6