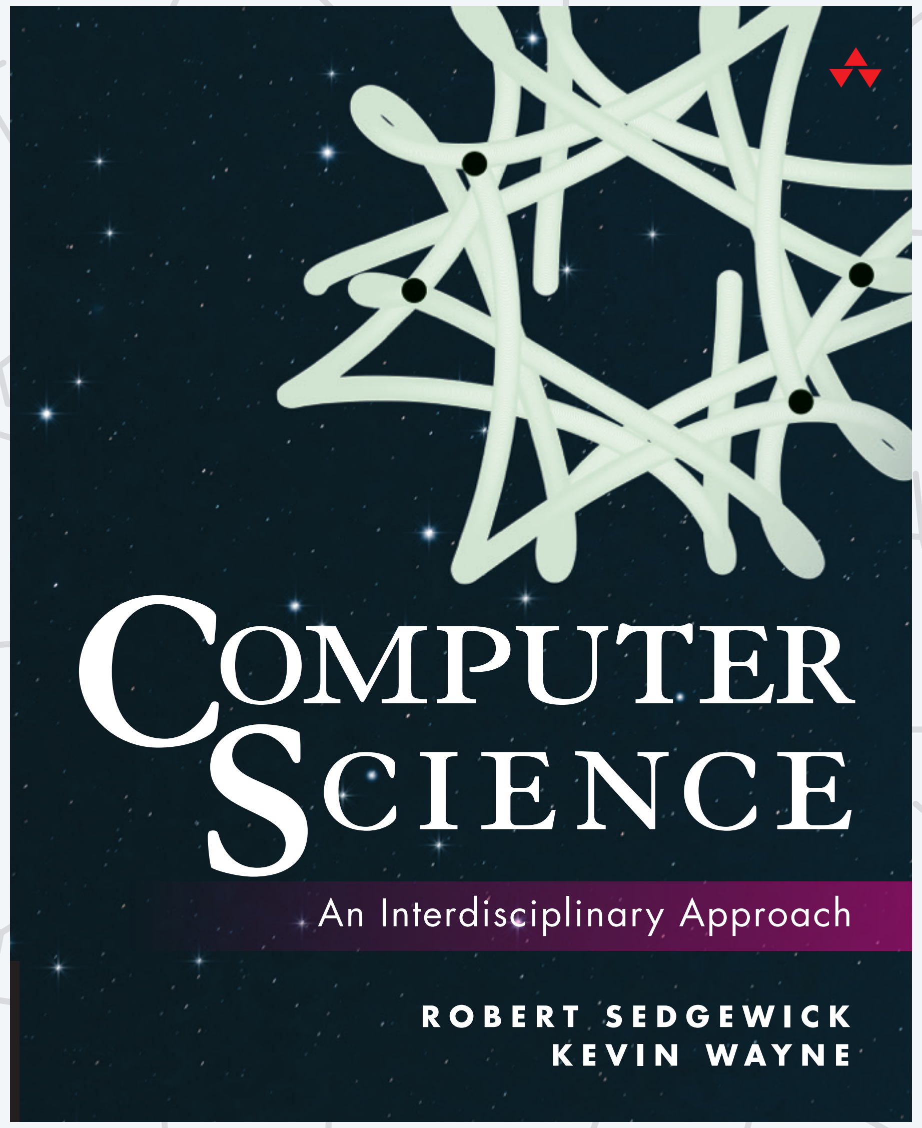
Computer Science

Robert Sedgewick | Kevin Wayne

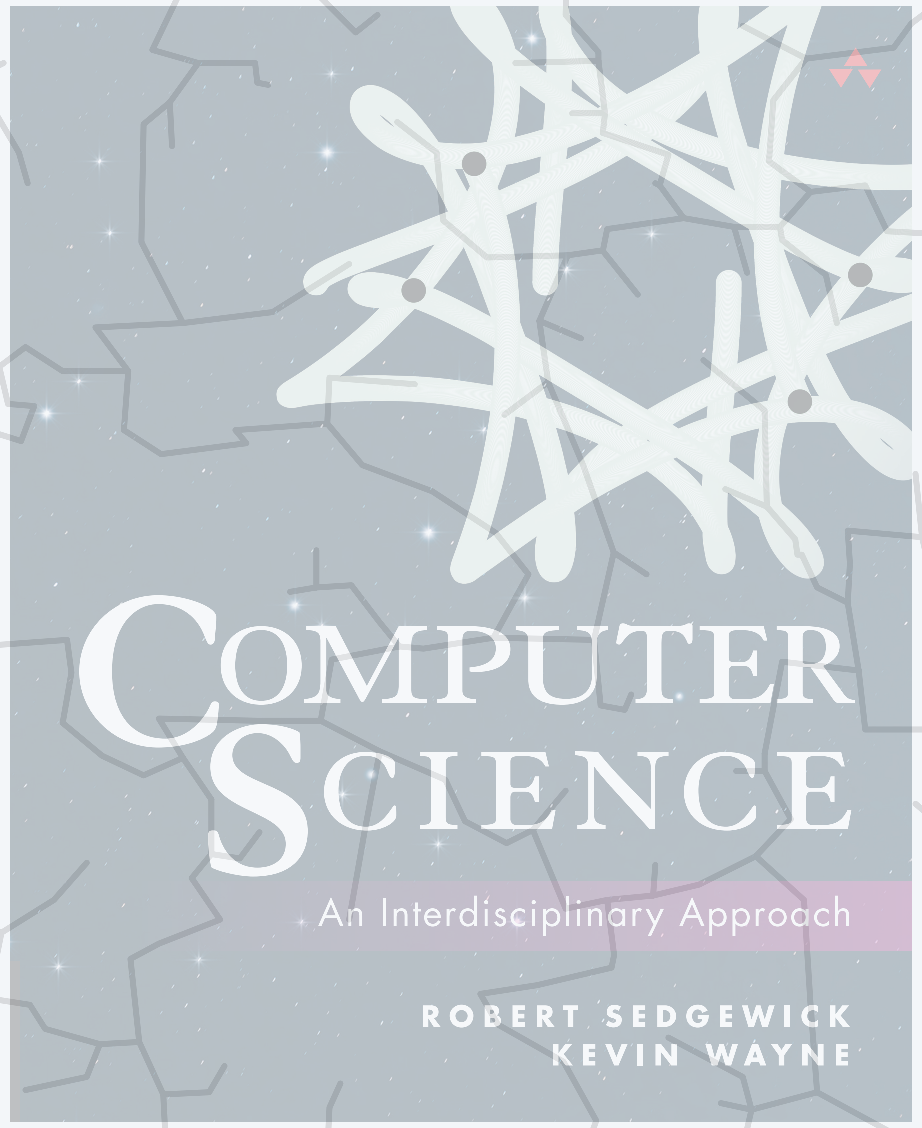


https://introcs.cs.princeton.edu

5. Theory of Computing

- *introduction*
- *models of computation*
- *universality*
- *computability*
- *halting problem*

Last updated on 4/5/25 6:54PM



https://introcs.cs.princeton.edu

5. Theory of Computing

Introduction to theory of computing

Fundamental questions.

- What is an algorithm?
- What is a general-purpose computer?
- What can/can't a computer do?
- What can/can't a computer do with limited resources?

History. Pioneering work at Princeton in the 1930s.

David Hilbert

Kurt Gödel

Alonzo Church

Alan Turing

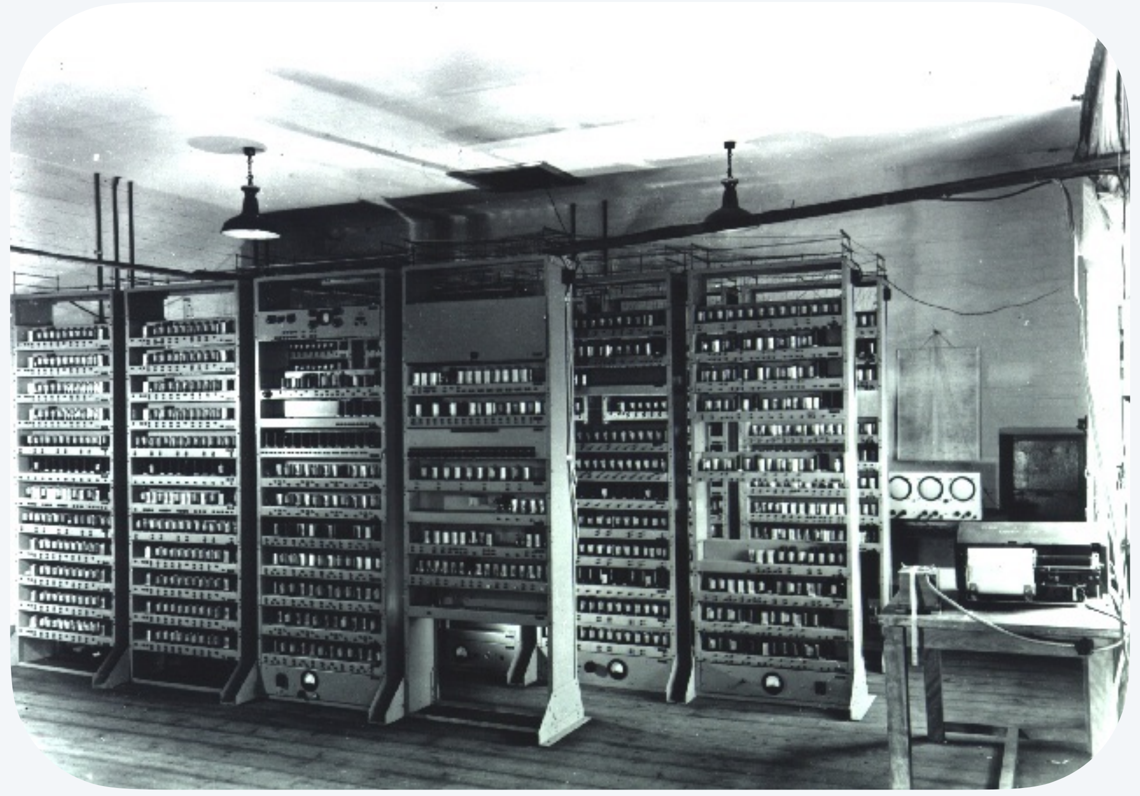
Introduction to theory of computing

Fundamental questions.

- What is an algorithm?
- What is a general-purpose computer?
- What can/can't a computer do?
- What can/can't a computer do with limited resources?

General approach. Consider minimal abstract machines.

Surprising outcome. Sweeping and relevant statements about all computers.



Some computational problems

Function problem. Compute a mathematical function. ← *input can be numbers, text, image, video, code, … (encoded in binary)*

input x → **function f** → *output $f(x)$*

problem	description	input	output
integer addition	given two integers x and y, what is $x + y$?	$1 + 2$	3
linear equation satisfiability	given a system of linear equations, does it have a solution?	$2a + 6b = 4$ $a + 3b = 3$	*no*
primality	given a positive integer x, is it prime?	17	*yes*
halting problem	given a function f and its input x, does the function halt on the given input?	`int x = 17;` `collatz(x);`	yes
⋮	⋮		

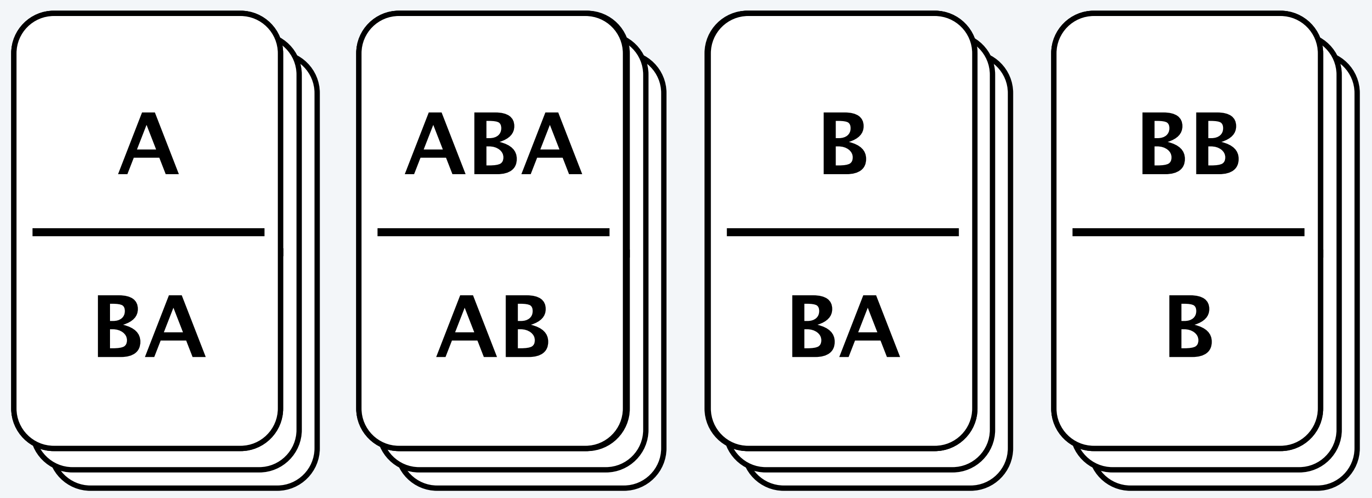
← *"decision problems" (output is yes/no)*

A warmup puzzle

Post's correspondence problem (PCP). Given n domino types, is there an arrangement of dominos with matching top and bottom strings?

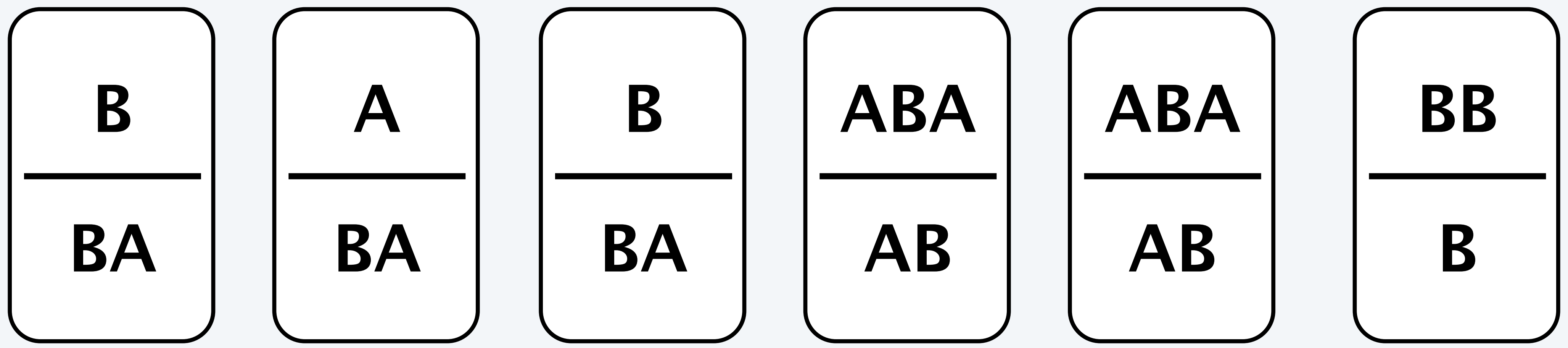
- Each domino has a top string and bottom string.
- No limit on the number of dominos used of each type.

Input.



0 1 2 3 $n = 4$

Solution. Yes.



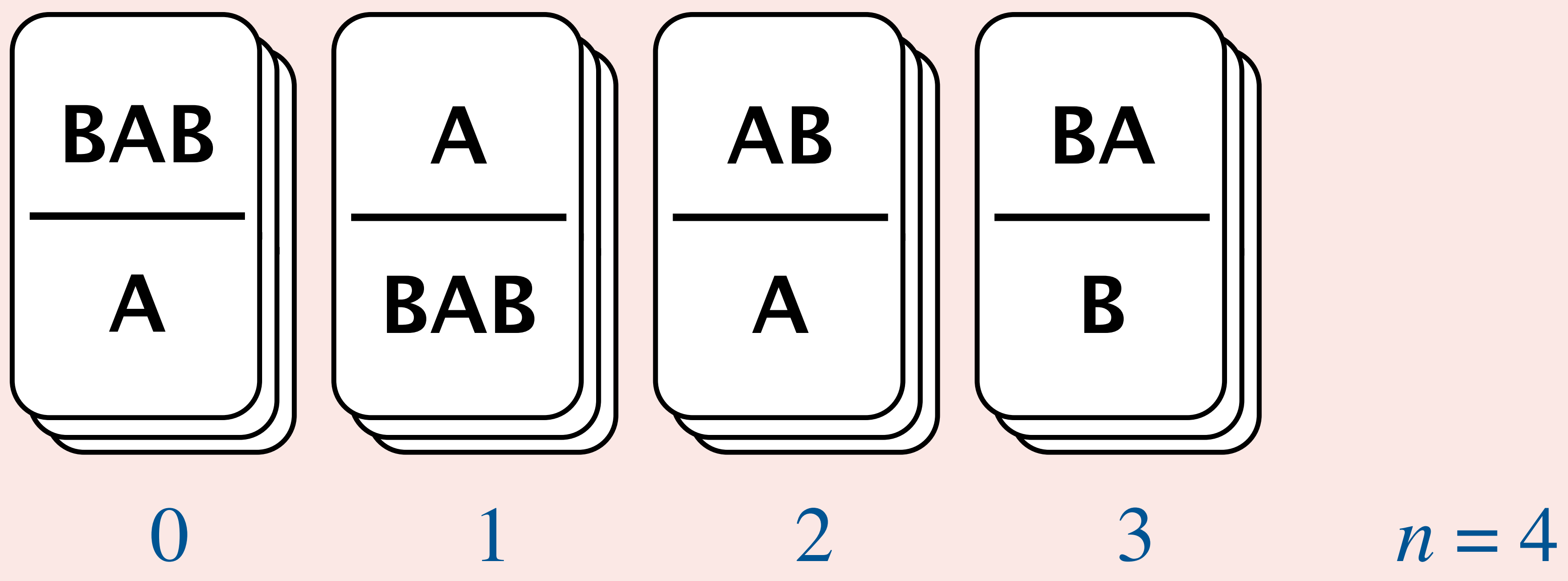
2 0 2 1 1 3

Theory of computing: quiz 1

Is there an arrangement of dominos with matching top and bottom strings?

A. Yes.

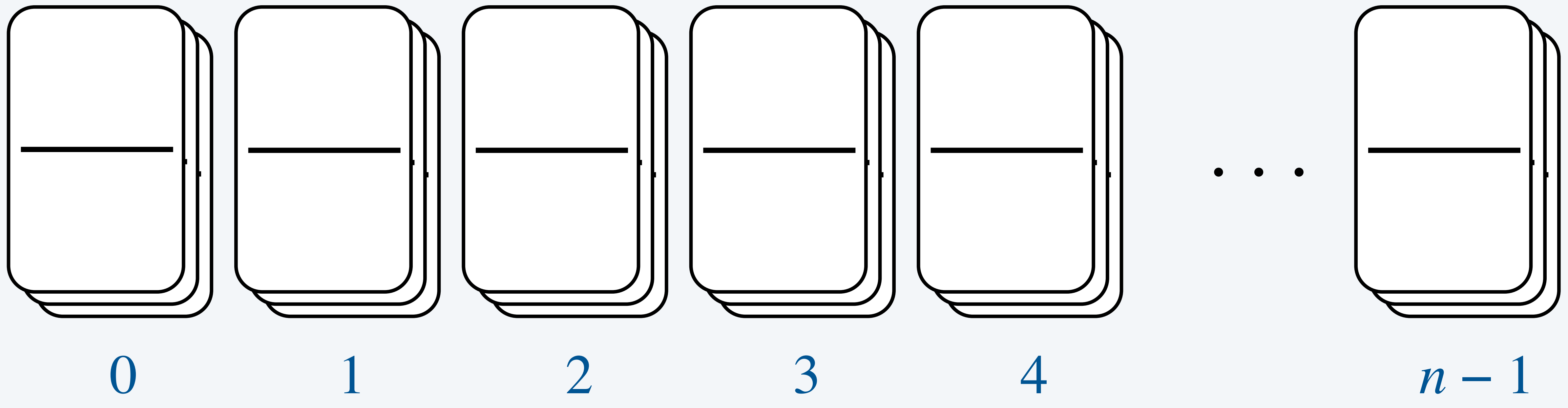
B. No.



A warmup puzzle

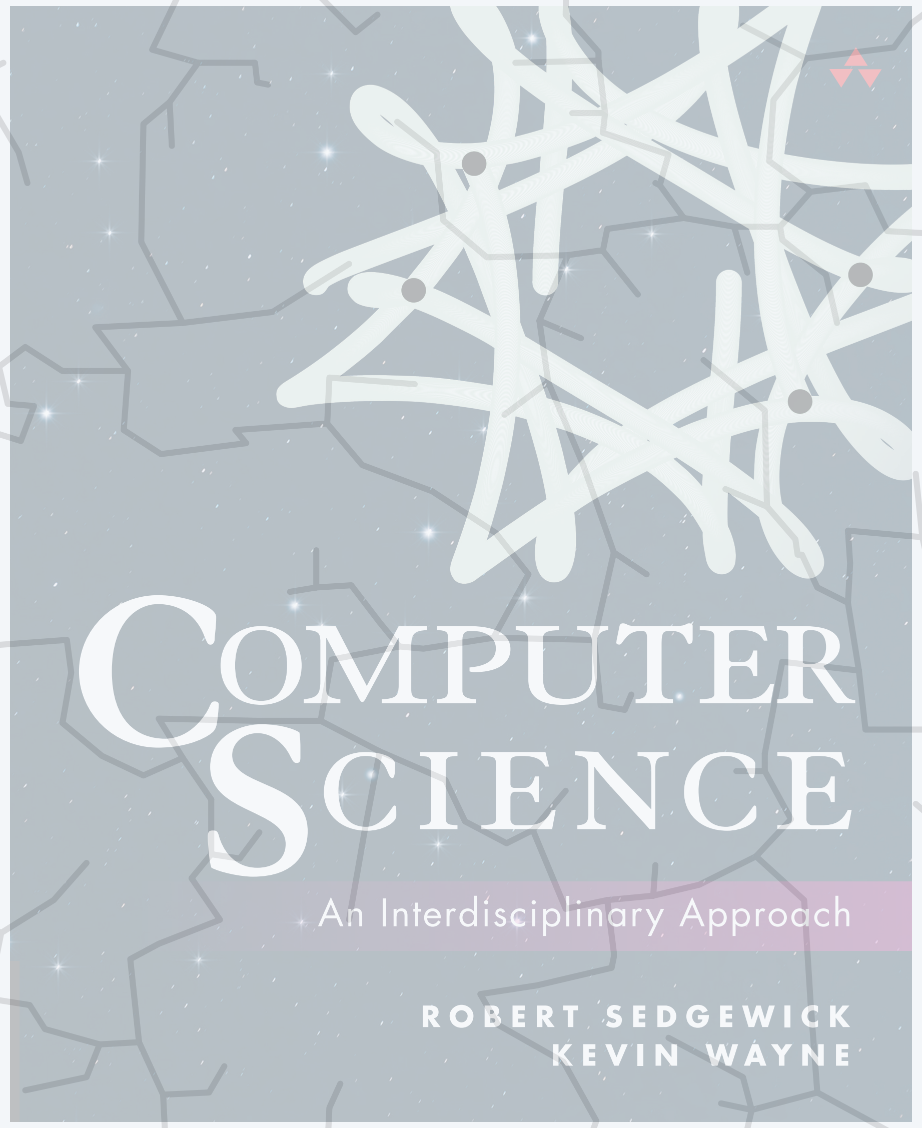
Post's correspondence problem (PCP). Given n domino types, is there an arrangement of dominos with matching top and bottom strings?

- Each domino has a top string and bottom string.
- No limit on the number of dominos used of each type.



A reasonable idea. Write a Java program that takes n domino types as input and solves PCP. ⟵ *but not so easy because you don't know how many dominos you will need*

Astonishing fact. It is provably impossible to write such a program!



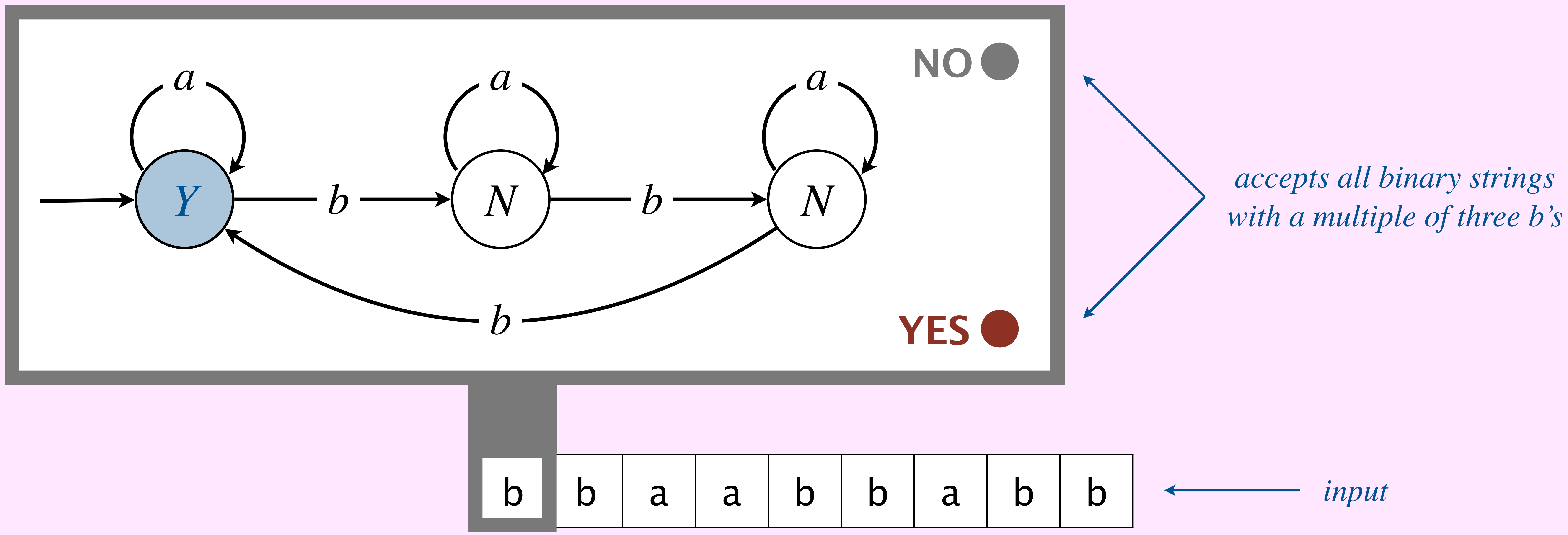
https://introcs.cs.princeton.edu

5. Theory of Computing

- introduction
- models of computation
- universality
- computability
- halting problem

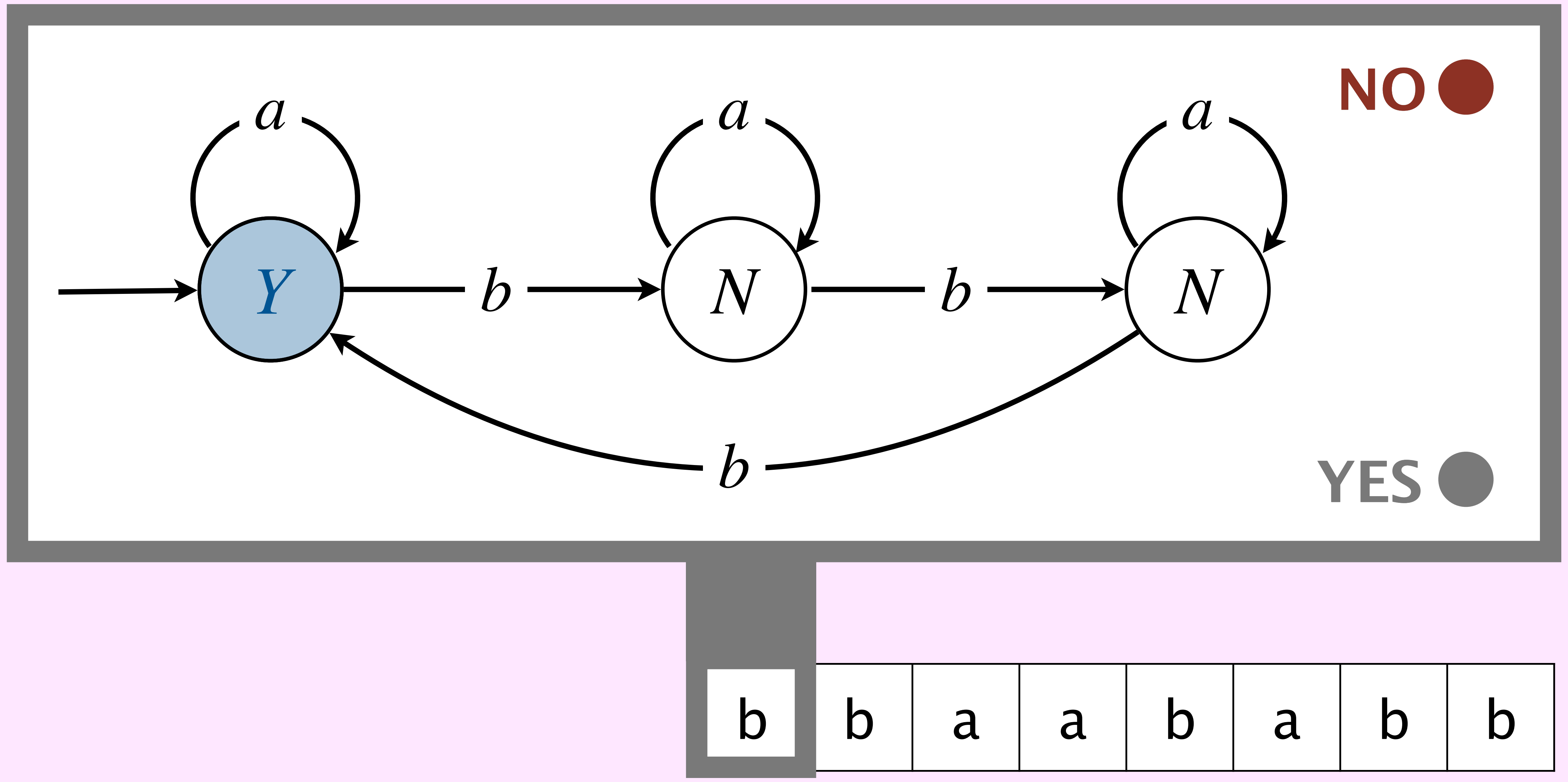
Deterministic finite-state automata demo

Goal. A simple model of computation.



Deterministic finite-state automata demo

Goal. A simple model of computation.



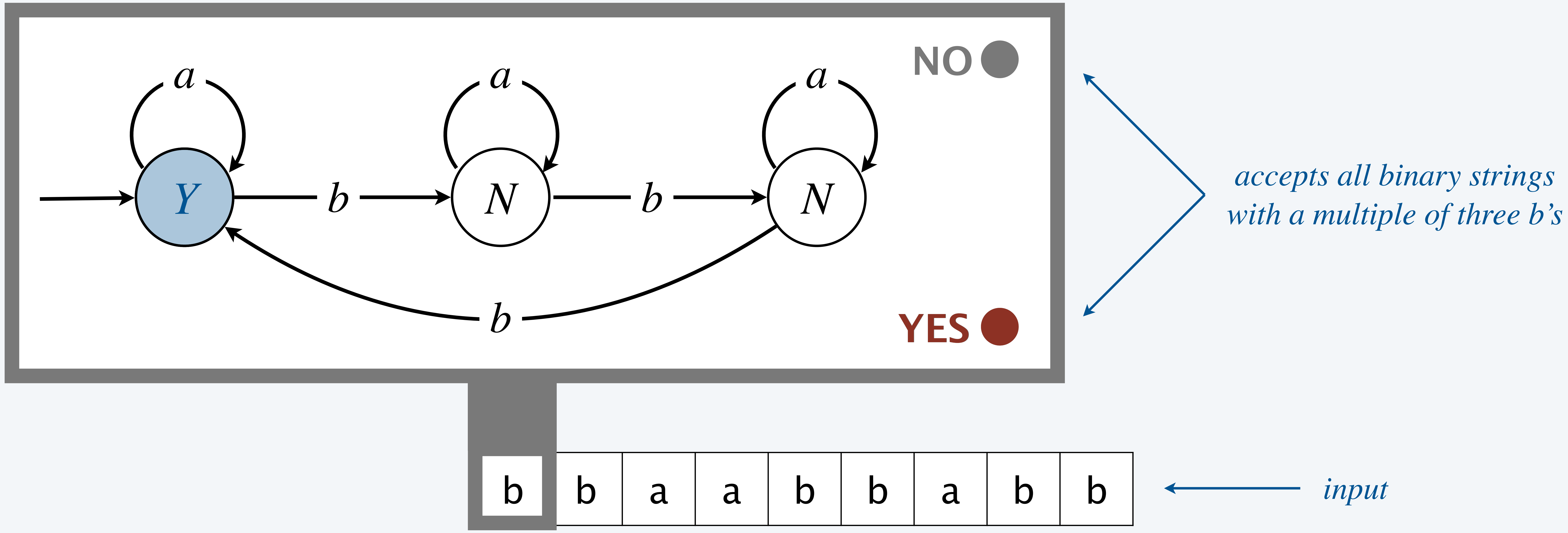
accepts all binary strings with a multiple of three b's

input

Deterministic finite-state automata

DFA. An abstract machine.

- Finite number of states.
- Begin in the start state; accept if end state is labeled Y.
- Repeat until the last input symbol has been consumed:
  - read next input symbol
  - move to the indicated state



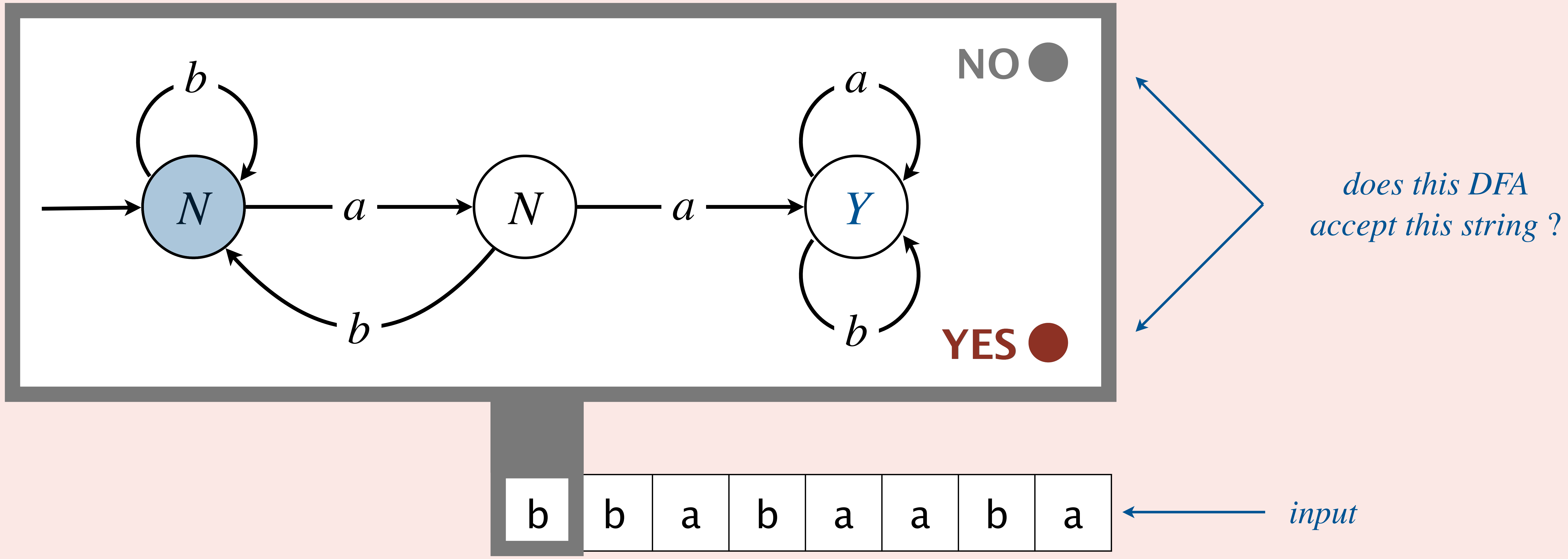
Describe the set of strings that the DFA matches.

A. All binary strings ending in aa.

B. All binary strings containing aa.

C. All binary strings containing at least two a's.

D. All binary strings containing an even number of a's.



Deterministic finite-state automata

Fact. DFAs can solve some important problems, but not others.

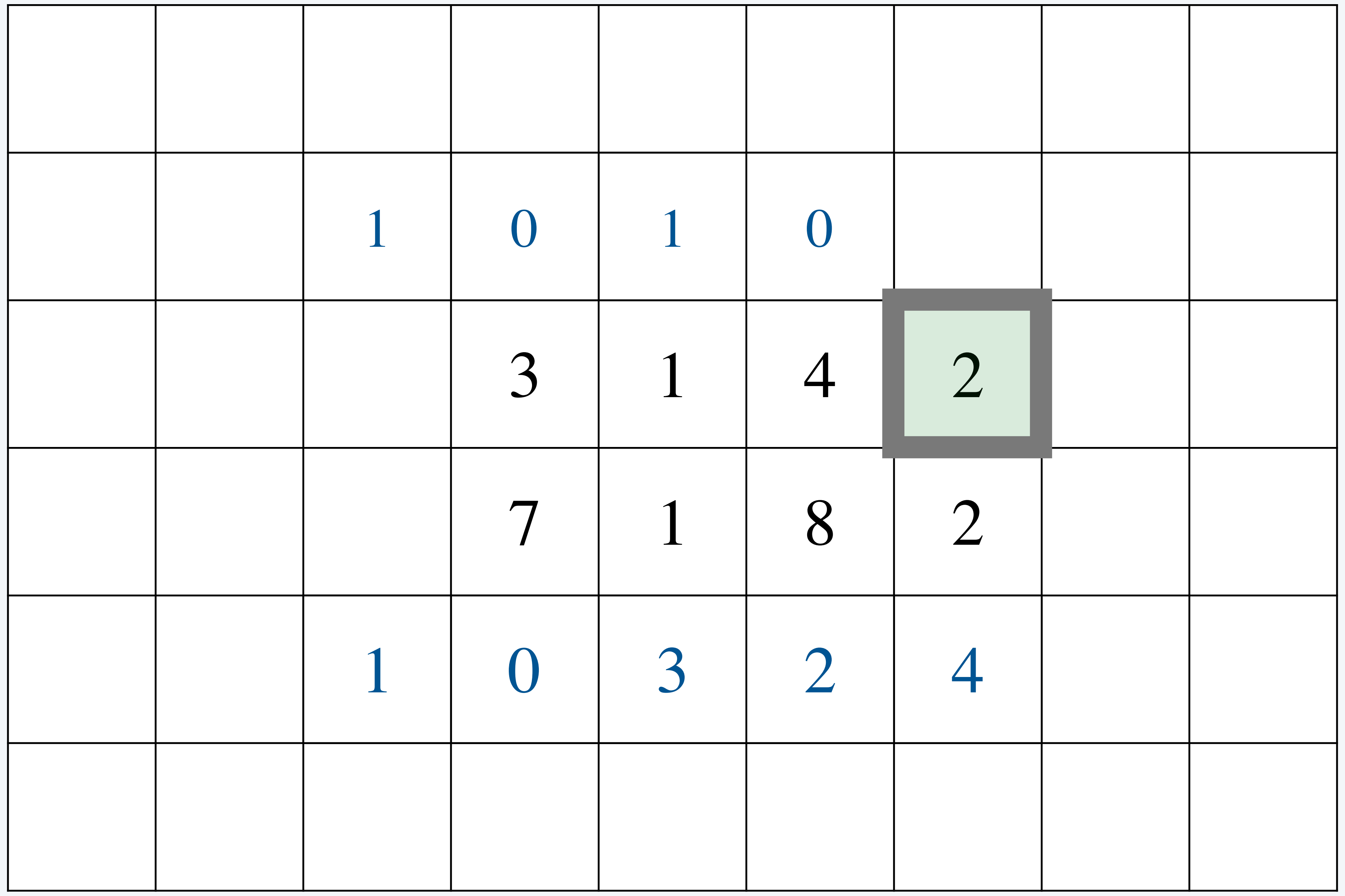
solvable with DFA	not solvable with DFA
even number of a's and b's	*equal number of a's and b's*
legal Java variable name	*legal Java program*
web form validation	*primality checking*
PROSITE pattern in genomics	*Watson–Crick palindrome*
sequential circuit	*Post's correspondence problem*
regular expression	*halting problem*
⋮	⋮

Turning machines: intuition

Goal. A simple model of computation that encompasses all known computational processes.

Approach. Characterize what a human "computer" can do with pencil, paper, and mechanical rules.

Ex. A familiar computational process.



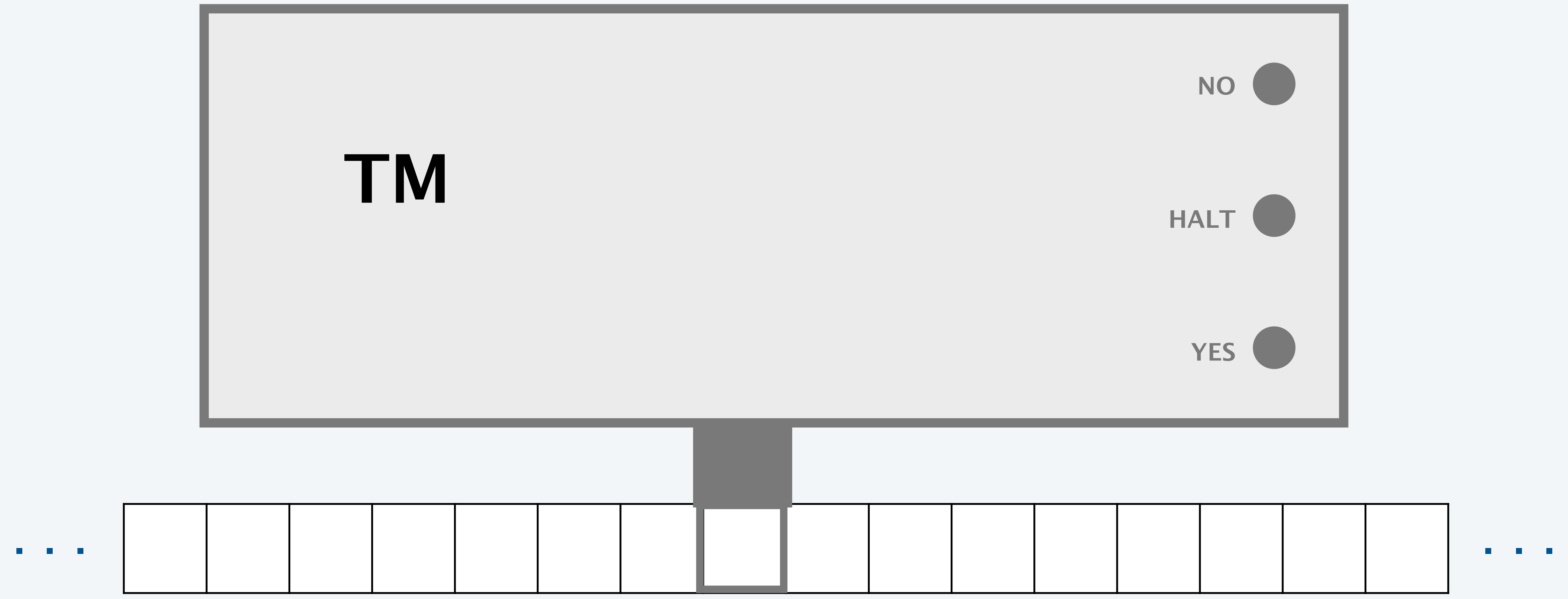
infinite loop possible

Key characteristics. Discrete; read/write; conditionals and loops; no prior limit on time/space.

Turing machines

Turing machine. An abstract machine that embodies mechanical rules on previous slide. ← *Turing gave precise mathematical description*

- Finite number of states and state transitions.
- Tape that stores symbols (for input, output, and intermediate results).
  - can read and write to tape
  - can move tape head left or right one cell
  - no limit on length



need separate TM for each task

Limitation. Each TM corresponds to one algorithm (or one program). Not programmable!

Universal Turing machine

Next goal. A “programmable” Turing machine.

Key insight. A TM can be represented as a string. ← *treat program as data*

Universal TM. A single TM that can compute anything computable by any TM.

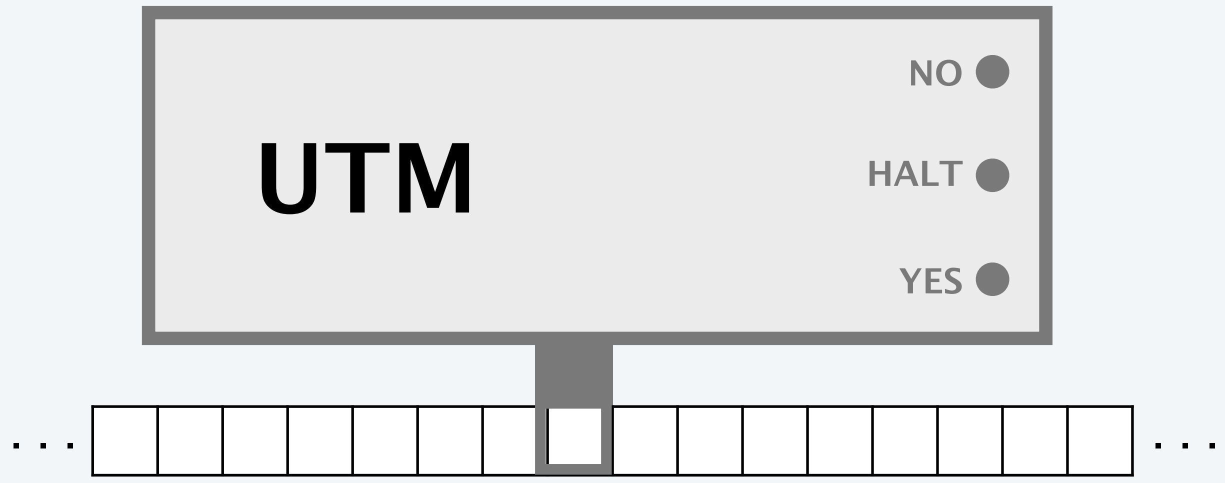
- Input: description of a TM and input for that TM.
- Output: the result of running that TM on that input.

A	B	A	A	B	⋆	3		0	1	#		0	2	R		0	0	1		0	1	#		1	2	1		1	0	#		H		2	2	2		0	1	#

TM input *separator* *description of Turing machine*

Theorem. [Turing 1936] There exists a universal TM.

Pf idea. Simulating a TM is a mechanical procedure.



Implications of universal Turing machine

TM. Formalizes the notion of an algorithm.

Universal TM. Formalizes the notion of a general-purpose computer.

we are so used to having a UTM in our pocket (smartphone), that we take this for granted

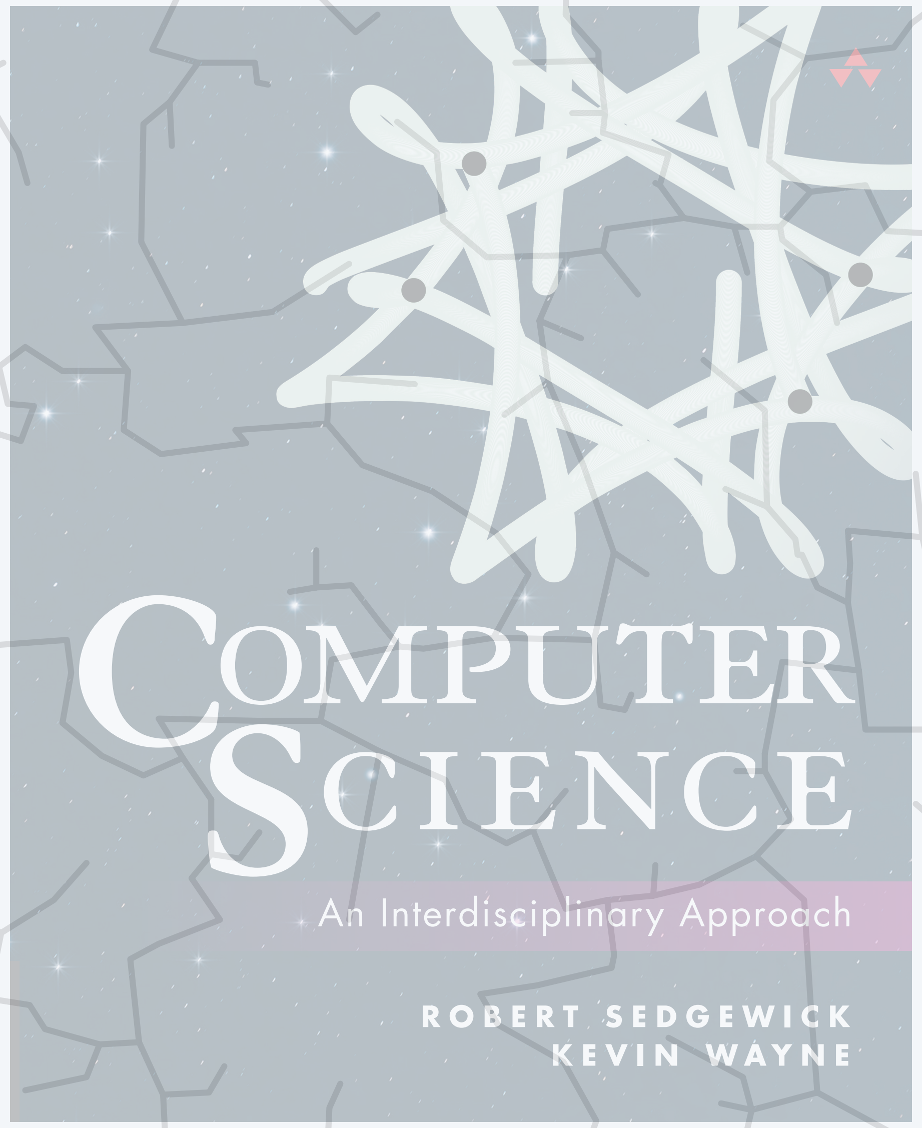
Profound implications.

for communication, photos, music, videos, games, calculators, word processing, …

- Single, universal, device.
- Anyone can invent a new way to use a computer.

pong, email, spreadsheet, web, search engine, e-commerce, social media, cryptocurrency, self-driving car, ChatGPT, …

> " *The importance of the universal machine is clear. We do not need to have an infinity of different machines doing different jobs…. The engineering problem of producing various machines for various jobs is replaced by the office work of 'programming' the universal machine.* " — *Alan Turing* (1948)



https://introcs.cs.princeton.edu

5. THEORY OF COMPUTING

- *introduction*
- *models of computation*
- *universality*
- *computability*
- *halting problem*

Church–Turing thesis

Church–Turing thesis. Any computational problem that can be solved by a physical system (in this universe) can be solved by a Turing machine.

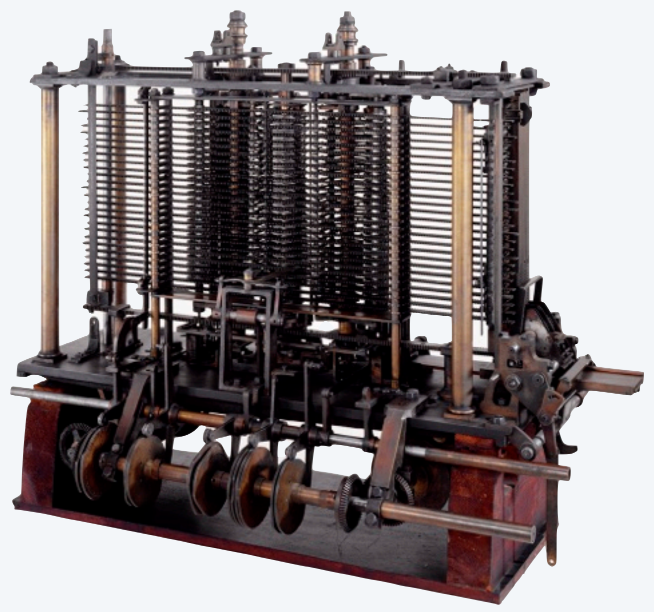
Remark. It's a thesis (not a theorem) since it's a statement about physics.

- Subject to falsification.
- Not subject to mathematical proof.

this is what we mean by "general-purpose computer"

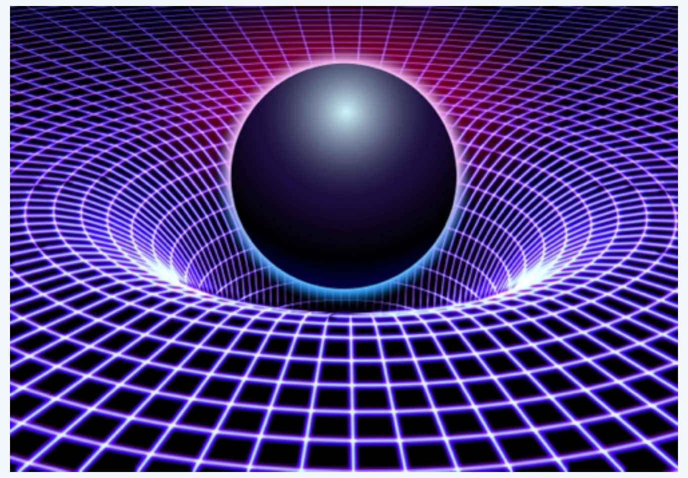
Implications.

- "All" computational devices can solve exactly the same computational problems.
- Turing's definition of computation is (equivalent to) the right one.
- Enables rigorous study of computation (in this universe).
- A new law of physics. (!)



Analytical Engine

DNA



black hole

Evidence supporting the Church–Turing thesis: random-access machines

Fact. All of these random-access machines are provably equivalent to a Turing machine.

- Macbook Pro, iPhone, Samsung Galaxy, supercomputer, …
- TOY machine. ← *stay tuned*
- …

← *ignoring limits of finite memory*




Implication 1. Processors are equivalent in terms of which computational problems they can solve.

Implication 2. Can't design processors that can solve more computational problems.

differences are in speed, power, cost, input/output, reliability, usability, …

Evidence supporting the Church–Turing thesis: programming languages

Fact. All of these programming languages are provably equivalent to a Turing-machine.

- Java.
- Python, C, C#, C++.
- Fortran, Lisp, Javascript, Matlab, R, Swift, Go, …
- …

ignoring intrinsic memory limitations

Implication 1. PLs are equivalent in terms of which computational problems they can solve.

Implication 2. Can't invent PL that can solve more computational problems.

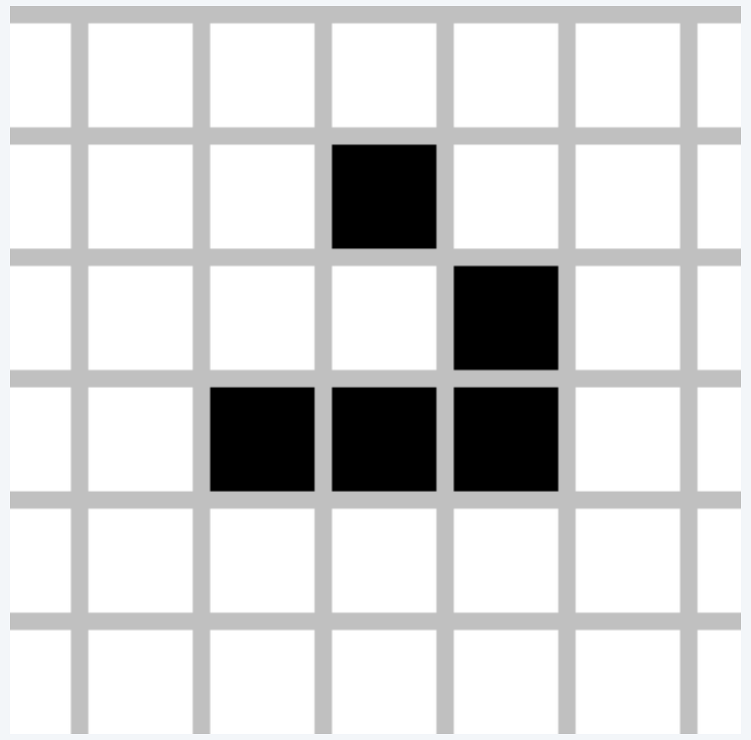
differences are in efficiency, writability, readability, maintainability, modularity, reliability, portability, and availability of libraries, …

More evidence supporting the Church–Turing thesis

Fact. All of these models of computation are provably equivalent to a Turing-machine.

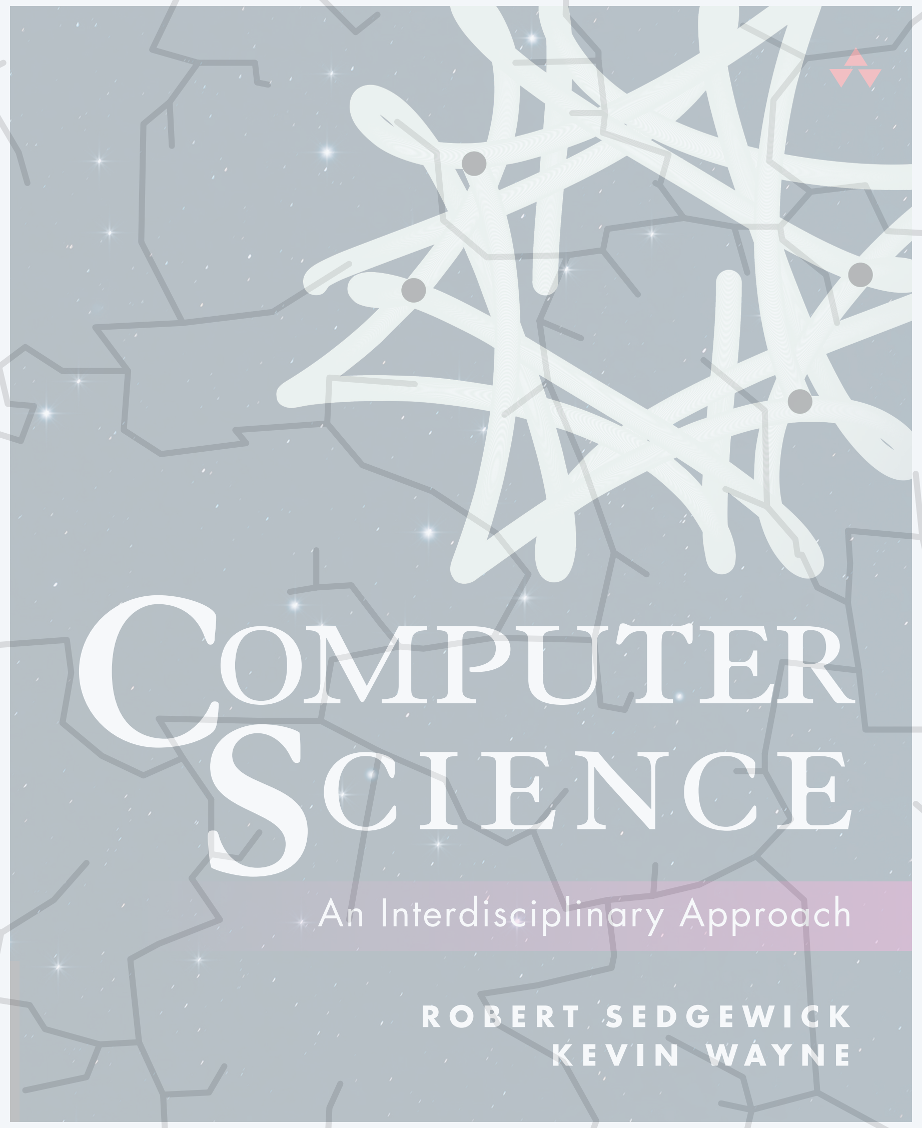
model of computation	description
programming languages	Java, Python, C, C#, C++, Fortran, Lisp, Javascript, …
random-access machines	Macbook Pro, iPhone, Samsung Galaxy, TOY, …
enhanced Turing machines	multiple heads, multiple tapes, 2D tape, nondeterminism
untyped λ-calculus	formal system for defining and manipulating functions
recursive functions	functions dealing with computation on integers
unrestricted grammars	iterative string replacement rules used by linguists
cellular automata	cells which change state based on local interactions
DNA computer	compute using biological operations on DNA
quantum computer	compute using superposition of quantum states
⋮	⋮

← *ignoring intrinsic memory limitations*



Which model of computation is not universal?

- **A.** Turing machines.
- **B.** DFAs.
- **C.** Java.
- **D.** iPhone 15 Pro.
- **E.** All of the above models are universal.



https://introcs.cs.princeton.edu

5. Theory of Computing

- *introduction*
- *models of computation*
- *universality*
- ***computability***
- *halting problem*

Computability

Def. A computational problem is computable if there exists a TM to solve it.

Def. A computational problem is uncomputable if no TM exists to solve it.

equivalently, Java program, iOS app, quantum computer, …

Theorem. [Turing 1936] The halting problem is uncomputable.

Theorem. [Post 1946] Post's correspondence problem is uncomputable.

Profound implications.

- There exist computational problems that no Turing machine can solve.
- There exist computational problems that no computer can solve.
- There exist computational problems that can't be solved in Java.

many such problems, and many that are important in practice

Implications for programming systems

Q. Why is debugging difficult?

A. All of the following computational problems are uncomputable.

problem	description
halting problem	Given a function f, does it halt on a given input x?
totality problem	Given a function f, does it halt on every input x?
no-input halting problem	Given a function f with no inputs, does it halt?
program equivalence	Do two function f and g always return the same value?
variable initialization	Is the variable x initialized before it is used?
dead-code elimination	Does this statement ever get executed?
memory management	Will an object x ever be referenced again?
⋮	⋮

UNCOMPUTABLE

Uncomputable problems from mathematics

Q. Why are some math calculations difficult?

A. The following computational problems are uncomputable.

UNCOMPUTABLE

problem	description	yes input	no input
Hilbert's 10^{th} problem	Given a polynomial equation with integer coefficients, does there exist an integer-valued solution?	$6x^3yz^2 + 3xy^2 - x^3 = 10$ $(x, y, z) = (5, 3, 0)$	$x^2 + y^2 = 3$
definite integration	Given a rational function $f(x)$ composed of polynomial and trigonometric functions, does the integral $\int_{-\infty}^{\infty} f(x)\, dx$ exist?	$\int_{-\infty}^{\infty} \frac{\cos x}{1 + x^2} dx$ $= \pi / e$	$\int_{-\infty}^{\infty} \frac{\cos x}{1 - x^2}\, dx$
⋮	⋮		

More uncomputable problems

Q. Why are so many disciplines difficult?

A. The following computational problems are uncomputable.

problem	description
polygonal tiling	Is it possible to tile the plane with copies of a given polygon?
spectral gap	Does a given quantum mechanical system have a spectral gap?
ray tracing	Will a light ray reach some final position in an optical system?
data compression	What is the shortest program that will produce a given string?
virus detection	Is a given computer program a virus?
dynamical systems	Is a generalized shift Φ chaotic?
network coding	Does a given network admit a coding scheme?
Magic	Does a given player have a winning strategy in a game of Magic?
⋮	⋮

UNCOMPUTABLE

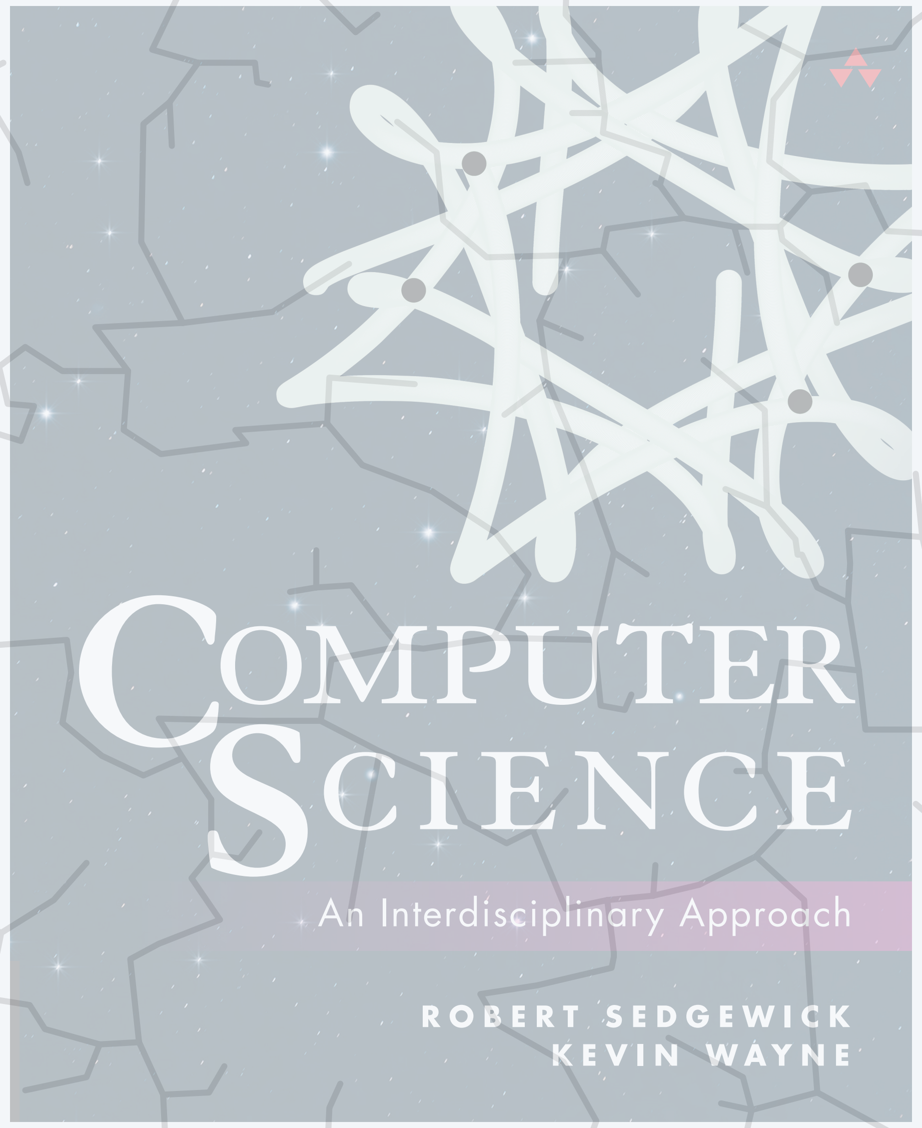
Which of these computational problems are computable?

A. Given a function f, determine whether it goes into an infinite loop.

B. Given a positive integer n, compute its integer factorization.

C. Both A and B.

D. Neither A nor B.



https://introcs.cs.princeton.edu

5. Theory of Computing

- introduction
- models of computation
- universality
- computability
- halting problem

The halting problem

Halting problem. Given a Java function `f()` and an input `x`, determine whether `f(x)` halts.

Ex. [Fermat's last theorem]

```
public static void f(int n) {
   for (int c = 1; true; c++)
      for (int a = 1; a <= c; a++)
         for (int b = 1; b <= c; b++)
            if (Math.pow(a, n) + Math.pow(b, n) == Math.pow(c, n))
               return;
}
```

assume arbitrary precision arithmetic (no overflow)

f(n) halts if and only if there are positive integers a, b, and c such that $a^n + b^n = c^n$

n	halts?	explanation
1	*yes*	$1^1 + 1^1 = 2^1$
2	*yes*	$3^2 + 4^2 = 5^2$
3	*no*	Euler 1760
4	*no*	Fermat 1670
5	*no*	Dirichlet, Legendre 1825
⋮	*no*	Wiles 1995

Ahead. It's impossible to write a Java program to solve the halting problem.

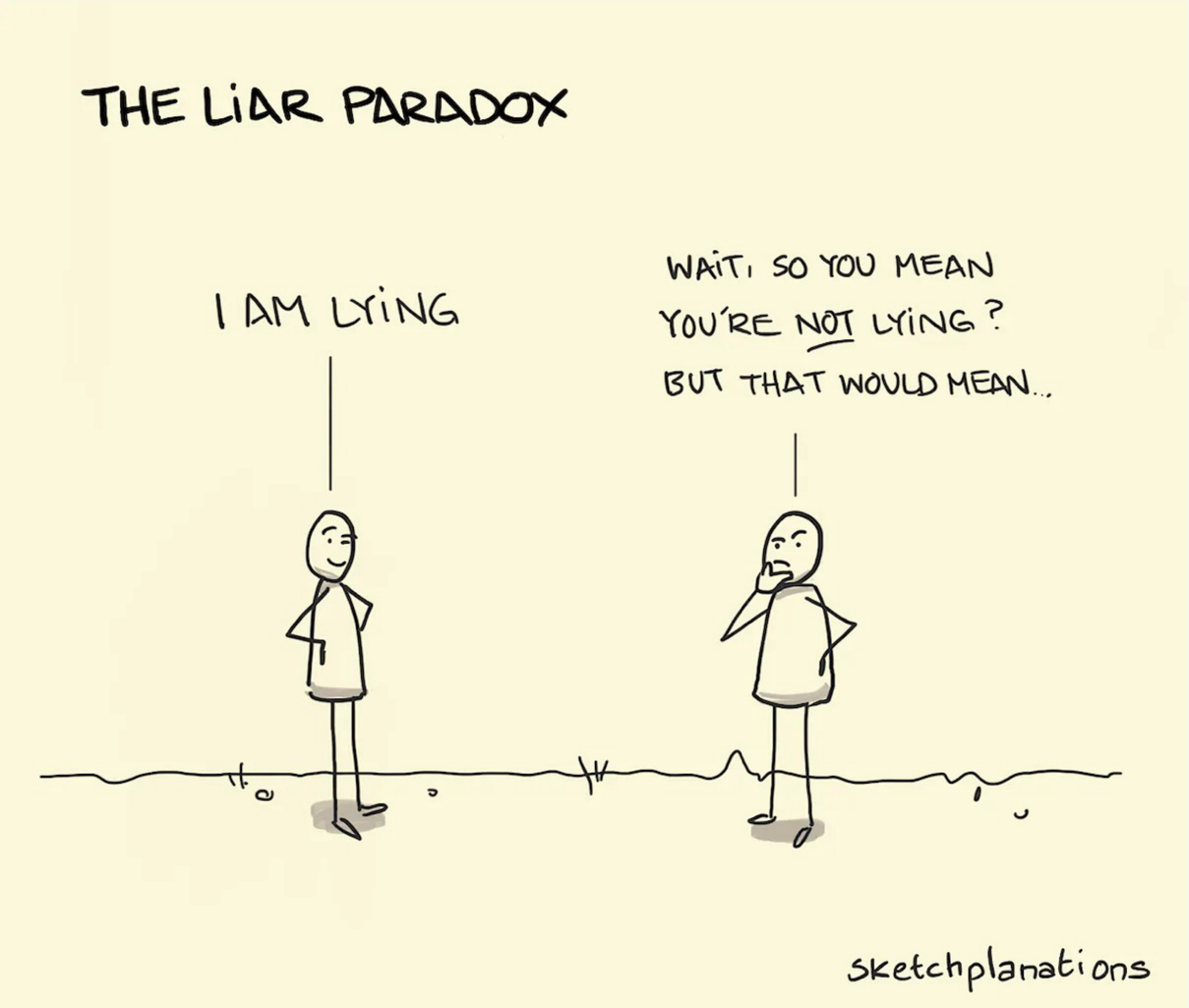
Crux of problem: can trace function on input n. If it halts, then you can safely conclude yes. But, if it does not seem to halt, then you don't know when to stop and conclude no.

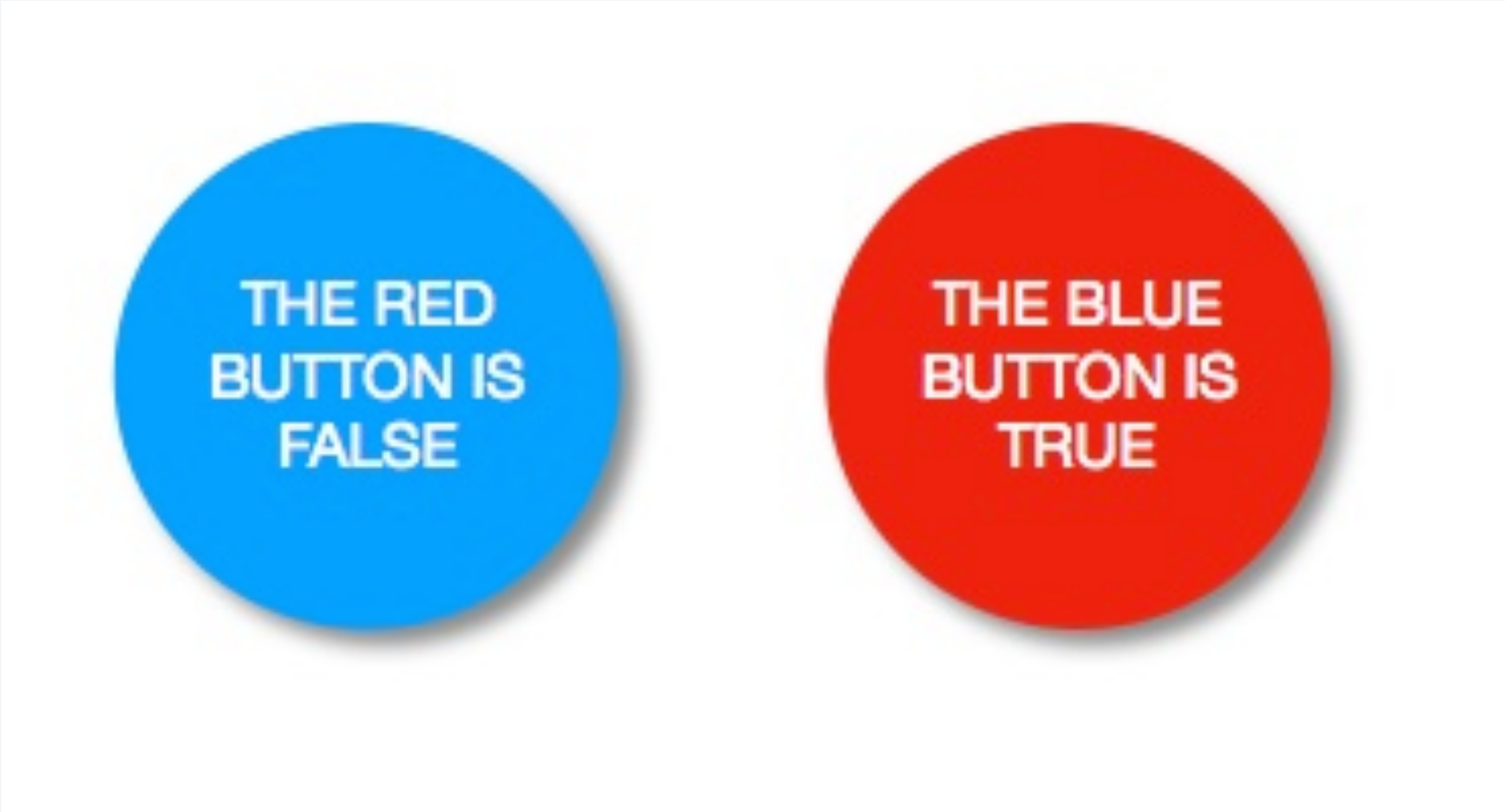
Note. Can solve the halting problem for some specific functions and/or inputs.

but that might be very very hard (even for 5-line Java functions)

Warmup: liar's paradox

Liar's paradox. [dates back to ancient Greek philosophers]





Logical conclusion. Cannot label all statements as *true* or *false*. ← *source of difficulty = self-reference*

The halting problem is uncomputable

Theorem. [Turing 1936] The halting problem is uncomputable.

Pf sketch. [by contradiction]

- Assume that there exists a function `halts()` that solves the halting problem. ← *Can assume it's in Java. Why?*

a function `f` *and its input* `x` *(both encoded as strings)*

purported solution to the halting problem

```
public boolean halts(String f, String x) {
   if ( /* f(x) halts */ ) return true;
   else                    return false;
}
```

`halt()` *returns either* `true` *or* `false` *(it cannot go into an infinite loop)*

Proof by contradiction: If a logical argument based on an assumption leads to a contradiction, then that assumption must have been *false*.

The halting problem is uncomputable

Theorem. [Turing 1936] The halting problem is uncomputable.

Pf sketch. [by contradiction]

- Assume that there exists a function `halts()` that solves the halting problem.

purported solution to the halting problem

```
public boolean halts(String f, String x) {
   if ( /* f(x) halts */ ) return true;
   else                    return false;
}
```

a client of halts()

```
public void strange(String f) {
   if (halts(f, f))
      while (true) { } // infinite loop
}
```

a contradiction?

```
strange(strange);
```

- Write a function `strange(f)` that goes into an infinite loop if `f(f)` halts; and halts otherwise.
- Call `strange()` with itself as argument. (!!)
  - ~~if `strange(strange)` halts, then `strange(strange)` goes into an infinite loop~~
  - ~~if `strange(strange)` does not halt, then `strange(strange)` halts~~
- This is a contradiction; therefore, `halts()` cannot exist. ▪

Big ideas

Turing machine. A, simple, formal model of computation.

Duality of programs and data. Encode both as strings and compute with both.

Universality. Concept of general-purpose programmable computers.

Church–Turing thesis. Computable at all = computable with a Turing machine.

Computability. There exist inherent limits to computation.

foundational ideas, all introduced in Turing's landmark paper

Turing's 1936 paper. One of the most impactful scientific papers of the 20^{th} century.

ON COMPUTABLE NUMBERS, WITH AN APPLICATION TO THE ENTSCHEIDUNGSPROBLEM

By A. M. TURING.

[Received 28 May, 1936.—Read 12 November, 1936.]

Credits

image	source	license
David Hilbert	Wikimedia	public domain
Kurt Gödel	Wikimedia	public domain
Alonzo Church	Princeton University	
Alan Turing	Science Museum, London	
EDSAC	Computer Laboratory, Cambridge	CC BY 2.0
Vintage Desktop Computer	Adobe Stock	education license
Macbook Pro M1	Apple	
Google Dalles Data Center	Google	
Theory vs. Practice	Ela Sjolie	
Sound Effects	Mixkit	Mixkit free license
Babbage's Analytical Engine	Science Museum, London	CC BY 2.0
DNA Computer	Clean Future	
Black Hole Gravity	Adobe Stock	education license

Credits

image	source	license
iPhone 14 Pro Max	Apple	
Samsung Galaxy Z	Samsung	
IBM Summit Supercomputer	Oak Ridge National Laboratories	
Quantum Computer	Erik Lucero / Google	
Conway's Game of Life	Wikimedia	CC BY-SA 3.0
Quantum Computing Logo	Adobe Stock	education license
DNA Computer	DNews	
Liar's Paradox	sketchplanations	
Red Button, Blue Button	Martin Svatoš	
Light Bulb Idea	Clker-Free-Vector-Images	Pixabay
On Computable Numbers	Alan Turing	