BBM 202 - ALGORITHMS

HACETTEPE UNIVERSITY

DEPT. OF COMPUTER ENGINEERING

INTRODUCTION TO UNDECIDABILITY

Acknowledgement: The course slides are adapted from the slides prepared by R. Sedgewick and K. Wayne of Princeton University.

Universality and computability

Fundamental questions
- What is a general-purpose computer?
- Are there limits on the power of digital computers?
- Are there limits on the power of machines we can build?

Pioneering work at Princeton in the 1930s.


David Hilbert
1862–1943

Asked the questions


Kurt Gödel
1906–1978

Solved the math problem


Alonzo Church
1903–1995

Solved the decision problem


Alan Turing
1912–1954

Provided THE answers

Context: Mathematics and logic

Mathematics. Any formal system powerful enough to express arithmetic.

Principia Mathematics
Peano arithmetic
Zermelo-Fraenkel set theory
.
.
.

Complete. *Can* prove truth or falsity of any arithmetic statement.
Consistent. *Cannot* prove contradictions like $2 + 2 = 5$.
Decidable. An algorithm exists to determine truth of every statement.

Q. (Hilbert, 1900) Is mathematics complete and consistent?
A. (Gödel's Incompleteness Theorem, 1931) NO (!!!)

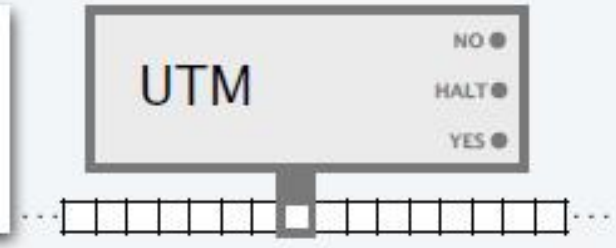
Q. (Hilbert's Entscheidungsproblem) Is mathematics decidable?
A. (Church 1936, Turing 1936) NO (!!)

Universality

UTM: A *simple* and *universal* model of computation.

Definition. A task is computable if a Turing machine exists that computes it.

Theorem (Turing, 1936). *It is possible to invent a single machine which can be used to do any computable task.*



Profound implications

- Any machine that can simulate a TM can simulate a universal Turing machine (UTM).
- Any machine that can simulate a TM can do *any* computable task.
- Don't need separate devices for solving scientific problems, playing music, email, . . .

A profound connection to the real world

Church-Turing thesis. Turing machines can do anything that can be described by *any* physically harnessable process of this universe: *All computational devices are equivalent.*

Remarks

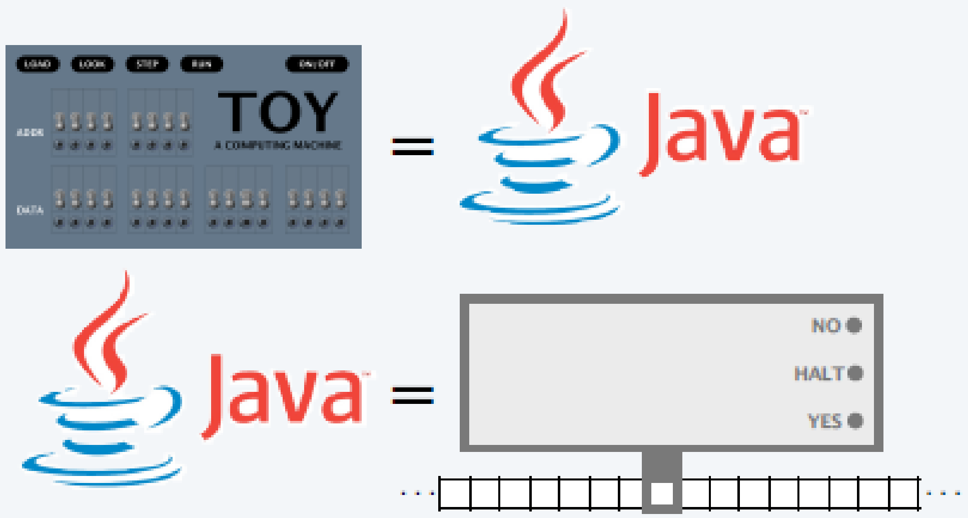
- A thesis, not a theorem.
- *Not* subject to proof.
- *Is* subject to falsification.

New model of computation or new physical process?

- Use *simulation* to prove equivalence.
- Example: TOY simulator in Java.
- Example: Java compiler in TOY.

Implications

- No need to seek more powerful machines or languages.
- Enables rigorous study of computation (in this universe).



Evidence in favor of the Church-Turing thesis

Evidence. Many, many models of computation have turned out to be equivalent (universal).

model of computation	*description*
enhanced Turing machines	multiple heads, multiple tapes, 2D tape, nondeterminism
untyped lambda calculus	method to define and manipulate functions
recursive functions	functions dealing with computation on integers
unrestricted grammars	iterative string replacement rules used by linguists
extended Lindenmayer systems	parallel string replacement rules that model plant growth
programming languages	Java, C, C++, Perl, Python, PHP, Lisp, PostScript, Excel
random access machines	registers plus main memory, e.g., TOY, Pentium
cellular automata	cells which change state based on local interactions
quantum computer	compute using superposition of quantum states
DNA computer	compute using biological operations on DNA
PCP systems	string matching puzzles (stay tuned)

8 decades without a counterexample, and counting.

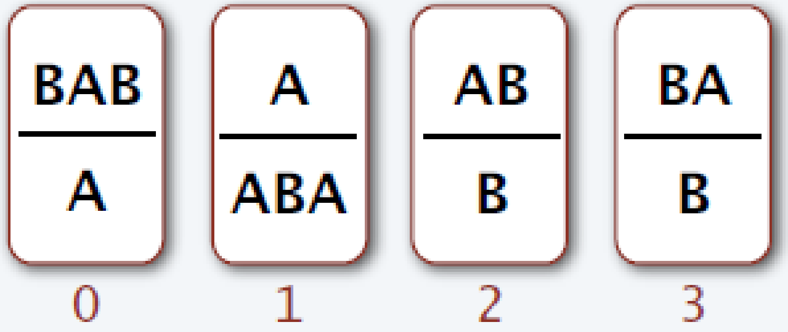
Post's correspondence problem (PCP)

PCP. A family of puzzles, each based on a set of cards.

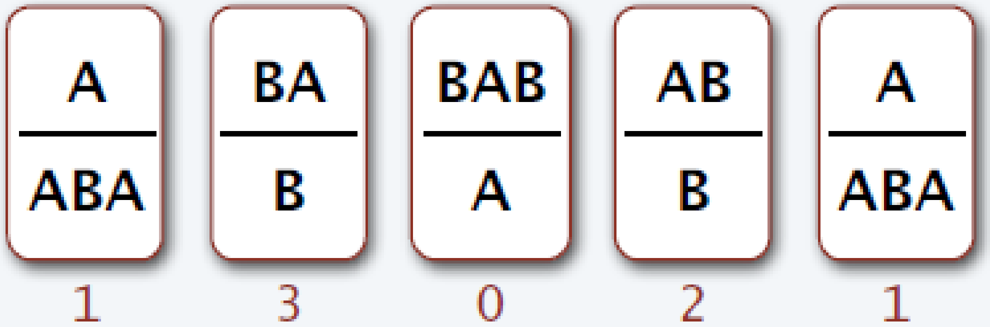
- *N* types of cards.
- No limit on the number of cards of each type.
- Each card has a top string and bottom string.

Does there exist an arrangement of cards with matching top and bottom strings?

Example 1 ($N = 4$).



Solution 1 (easy): YES.



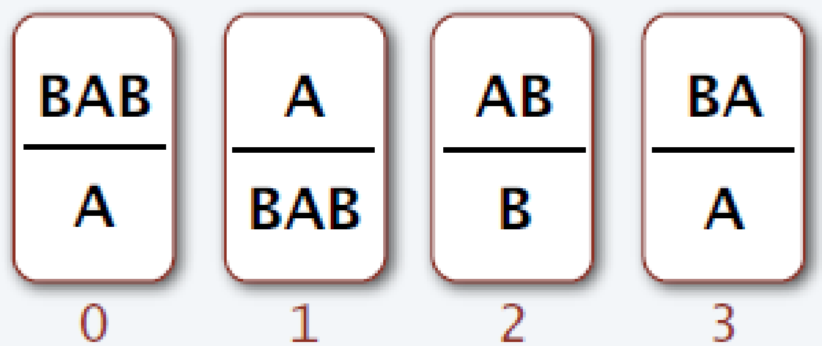
Post's correspondence problem (PCP)

PCP. A family of puzzles, each based on a set of cards.

- *N* types of cards.
- No limit on the number of cards of each type.
- Each card has a top string and bottom string.

Does there exist an arrangement of cards with matching top and bottom strings?

Example 2 ($N = 4$).



Solution 2 (easy): NO. No way to match even the first character!

Post's correspondence problem (PCP)

PCP. A family of puzzles, each based on a set of cards.

- *N* types of cards.
- No limit on the number of cards of each type.
- Each card has a top string and bottom string.

Does there exist an arrangement of cards with matching top and bottom strings?

Example 3 (created by Andrew Appel).

0	1	2	3	4	5	6	7	8	9	10
S[	X	BAB	11A	1	[A	]	[	B1	B]	[1A]E
S[11111X][	1X	A	A1	1	[B	]	[	1B	A]	E

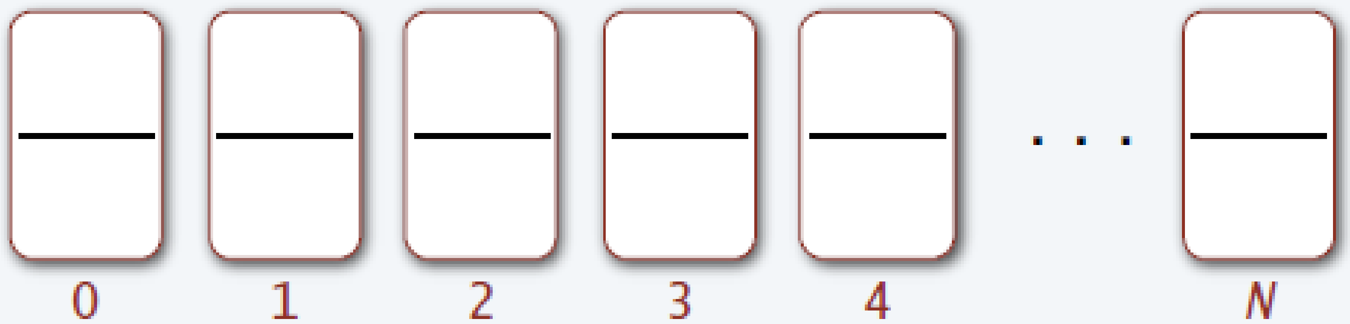
Challenge for the bored: Find a solution that starts with a card of type 0.

Post's correspondence problem (PCP)

PCP. A family of puzzles, each based on a set of cards.

- N types of cards.
- No limit on the number of cards of each type.
- Each card has a top string and bottom string.

Does there exist an arrangement of cards with matching top and bottom strings?



A reasonable idea. Write a program to take N card types as input and solve PCP.

A surprising fact. It is *not possible* to write such a program.

Another impossible problem

Halting problem. Write a Java program that reads in code for a Java static method f() and an input x, and decides whether or not f(x) results in an infinite loop.

Example 1 (easy).

```
public void f(int x)
{
   while (x != 1)
   {
      if  (x % 2 == 0) x = x / 2;
      else             x = 2*x + 1;
   }
}
```

Halts only if x is a positive power of 2

Example 2 (difficulty unknown).

```
public void f(int x)
{
   while (x != 1)
   {
      if  (x % 2 == 0) x = x / 2;
      else             x = 3*x + 1;
   }
}
```

Involves *Collatz conjecture* (see Recursion lecture)

```
f(7):     7 22 11 34 17 52 26 13 40 20 10 5 16 8 4 2 1
f(-17): -17 -50 -25 -74 -37 -110 -55 -164 -82 -41 -122 ... -17 ...
```

Next. A proof that it is *not possible* to write such a program.

Undecidability of the halting problem

Definition. A yes-no problem is undecidable if no Turing machine exists to solve it.
(A problem is computable if a Turing machine does exist that solves it.)

Theorem (Turing, 1936). The halting problem is undecidable.

Profound implications

- There exists a problem that no Turing machine can solve.
- There exists a problem that no *computer* can solve.
- There exist *many problems* that no computer can solve (stay tuned).

Warmup: self-referential statements

Liar paradox (dates back to ancient Greek philosophers).

- Divide all statements into two categories: *true* and *false*.
- Consider the statement "This statement is *false*."
- Is it *true*? If so, then it is *false*, a contradiction.
- Is it *false*? If so, then it is *true*, a contradiction.

Source of the difficulty: Self-reference.

true	*false*
$2 + 2 = 4$	$2 + 2 = 99$
The earth is round.	The earth is flat.
Starfish have no brains.	Earthworms have 3 hearts.
Venus rotates clockwise.	Saturn rotates clockwise.
...	...
This statement is false. ✗	This statement is false. ✗

Logical conclusion. Cannot label *all* statements as *true* or *false*.

Proof of the undecidability of the halting problem

Theorem (Turing, 1936). The halting problem is undecidable.

Proof outline.

- Assume the existence of a function `halt(f, x)` that solves the problem.

```
public boolean halt(String f, String x)
{
   if ( /* something terribly clever */ ) return true;
   else                                   return false;
}
```

By universality, may as well use Java.
(If this exists, we could simulate it on a TM.)

- Arguments: A function `f` and input `x`, encoded as strings.
- Return value: `true` if `f(x)` halts and `false` if `f(x)` does not halt.
- `halt(f, x)` always halts.
- Proof idea: *Reductio ad absurdum*: if any logical argument based on an assumption leads to an absurd statement, then the assumption is false.

Proof of the undecidability of the halting problem

Theorem (Turing, 1936). The halting problem is undecidable.

Proof.

- Assume the existence of a function `halt(f, x)` that solves the problem.
- Create 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.
- *Reductio ad absurdum.*
- `halt(f, x)` cannot exist.

Solution to the problem

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

A client

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

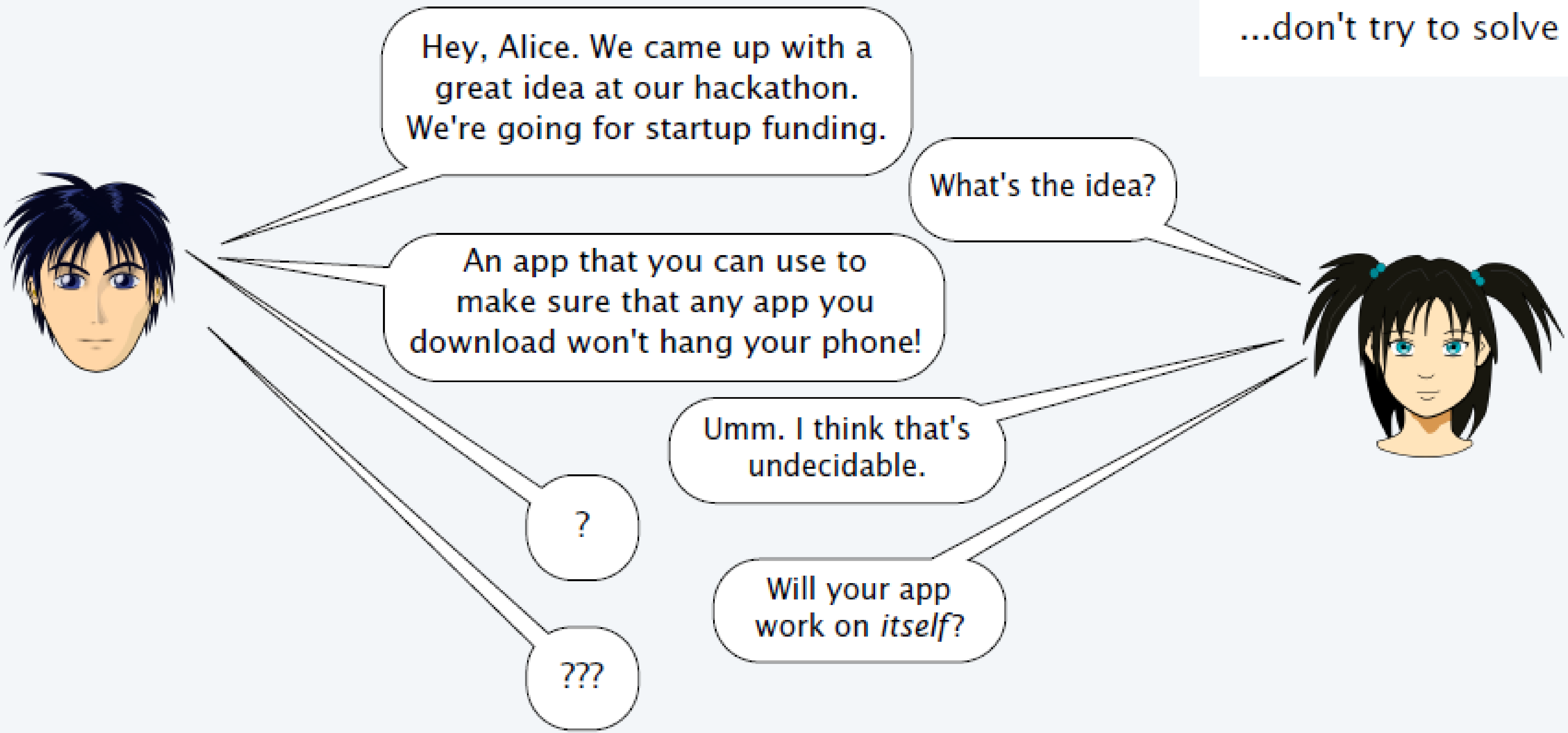
A contradiction

```
strange(strange)
```

halts?
does not halt?

Implications of undecidability

Primary implication. If you know that a problem is undecidable...



...don't try to solve it!

Implications for programming systems

Q. Why is debugging difficult?
A. All of the following are *undecidable*.

Halting problem. Give a function f, does it halt on a given input x?
Totality problem. Give a function f, does it halt on *every* input x?
No-input halting problem. Give a function f with no input, does it halt?
Program equivalence. Do two functions f and g always return same value?
Uninitialized variables. Is the variable x initialized before it's used?
Dead-code elimination. Does this statement ever get executed?

UNDECIDABLE

Prove each by reduction from the halting problem: A solution would solve the halting problem.

Q. Why are program development environments complicated?
A. They are programs that manipulate programs.

Another undecidable problem

The Entscheidungsproblem (Hilbert, 1928) ← "Decision problem"

- Given a first-order logic with a finite number of additional axioms.
- Is the statement provable from the axioms using the rules of logic?

UNDECIDABLE

David Hilbert
1862–1943

Lambda calculus

- Formulated by Church in the 1930s to address the Entscheidungsproblem.
- Also the basis of modern functional languages.

HASKELL JAVA

Alonso Church
1903–1995

Theorem (Church and Turing, 1936). The Entscheidungsproblem is undecidable.

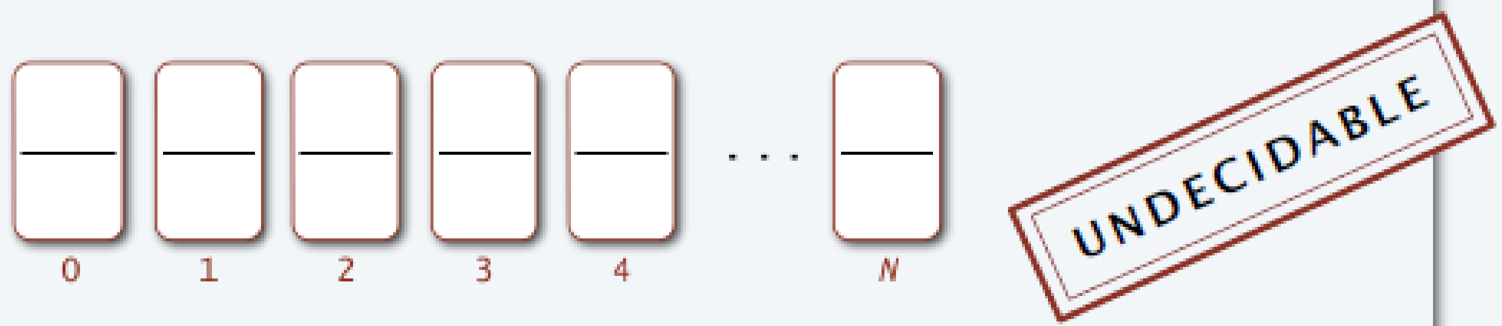
Another undecidable problem

Post's correspondence problem (PCP)

PCP. A family of puzzles, each based on a set of cards.

- *N* types of cards.
- No limit on the number of cards of each type.
- Each card has a top string and bottom string.

Does there exist an arrangement of cards with matching top and bottom strings?



A reasonable idea. Write a program to take *N* card types as input and solve PCP.

Theorem (Post, 1946). Post's correspondence problem is undecidable.

Examples of undecidability from computational mathematics

Hilbert's 10th problem

- Given a multivariate polynomial $f(x, y, z, \ldots)$.
- Does f have integral roots ? (Do there exist integers x, y, z, such that $f(x, y, z, \ldots) = 0$?)

UNDECIDABLE

Ex. 1 $f(x, y, z) = 6x^3yz^2 + 3xy^2 - x^3 - 10$

YES $f(5, 3, 0) = 0$

Ex. 2 $f(x, y) = x^2 + y^2 - 3$ NO

Definite integration

- Given a rational function $f(x)$ composed of polynomial and trigonometric functions.
- Does $\int_{-\infty}^{\infty} f(x)dx$ exist?

UNDECIDABLE

Ex. 1 $\frac{\cos(x)}{1+x^2}$ YES $\int_{-\infty}^{\infty} \frac{\cos(x)}{1+x^2}dx = \frac{\pi}{e}$

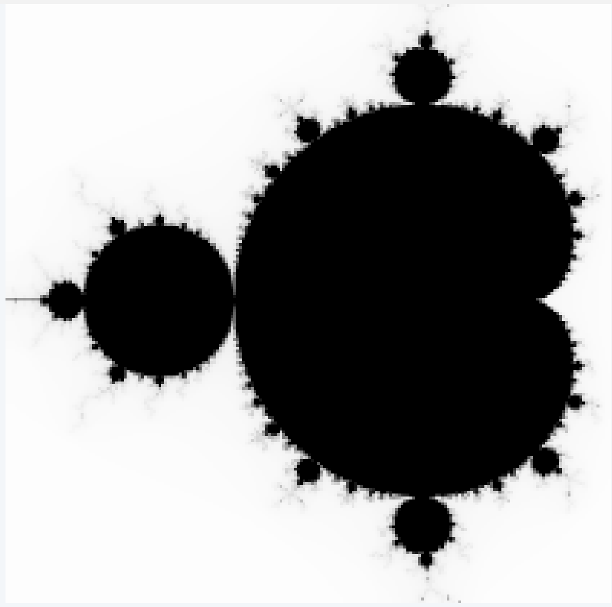
Ex. 2 $\frac{\cos(x)}{1-x^2}$ NO

Examples of undecidability from computer science

Optimal data compression

- Find the shortest program to produce a given string.
- Find the shortest program to produce a given *picture*.

UNDECIDABLE



produced by a 34-line Java program

Virus identification

- Is this code equivalent to this known virus?
- Does this code contain a virus?

UNDECIDABLE

```
Private Sub AutoOpen()
On Error Resume Next
If System.PrivateProfileString("", CURRENT_USER\Software
\Microsoft\Office\9.0\Word\Security",
                                "Level") <> "" Then
CommandBars("Macro").Controls("Security...").Enabled = False
. . .
For oo = 1 To AddyBook.AddressEntries.Count
   Peep = AddyBook.AddressEntries(x)
   BreakUmOffASlice.Recipients.Add Peep
   x = x + 1
   If x > 50 Then oo = AddyBook.AddressEntries.Count
Next oo
. . .
BreakUmOffASlice.Subject = "Important Message From " &
Application.UserName
BreakUmOffASlice.Body = "Here is that document you asked for
... don't show anyone else ;-)"
. . .
```

Melissa virus (1999)

Turing's key ideas

Turing's paper in the *Proceedings of the London Mathematical Society* "On Computable Numbers, With an Application to the Entscheidungsproblem" was one of the most impactful scientific papers of the 20th century.

Alan Turing
1912–1954

The Turing machine. A formal model of computation.

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

Universality. Concept of general-purpose programmable computers.

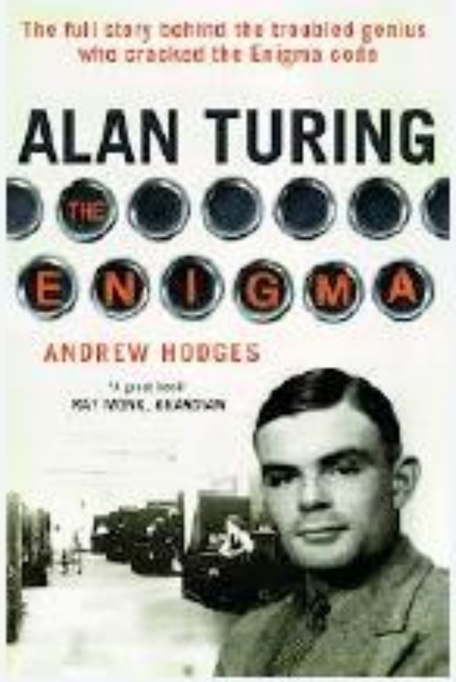
Church-Turing thesis. If it is computable at all, it is computable with a Turing machine.

Computability. There exist inherent limits to computation.

Turing's paper was published in 1936, *ten years before* Eckert and Mauchly worked on ENIAC (!)

Alan Turing: the father of computer science

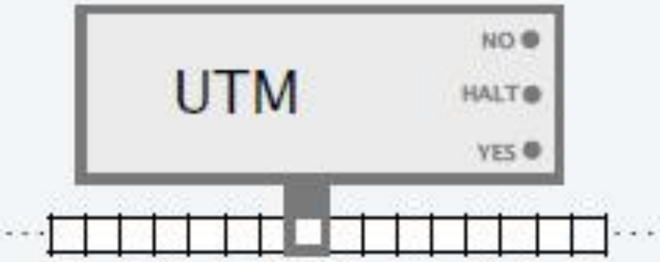
> *It was not only a matter of abstract mathematics, not only a play of symbols, for it involved thinking about what people did in the physical world.... It was a play of imagination like that of Einstein or von Neumann, doubting the axioms rather than measuring effects.... What he had done was to combine such a naïve mechanistic picture of the mind with the precise logic of pure mathematics. His machines – soon*



— *John Hodges, in* Alan Turing, the Enigma

A Google data center

=



A Universal Turing Machine