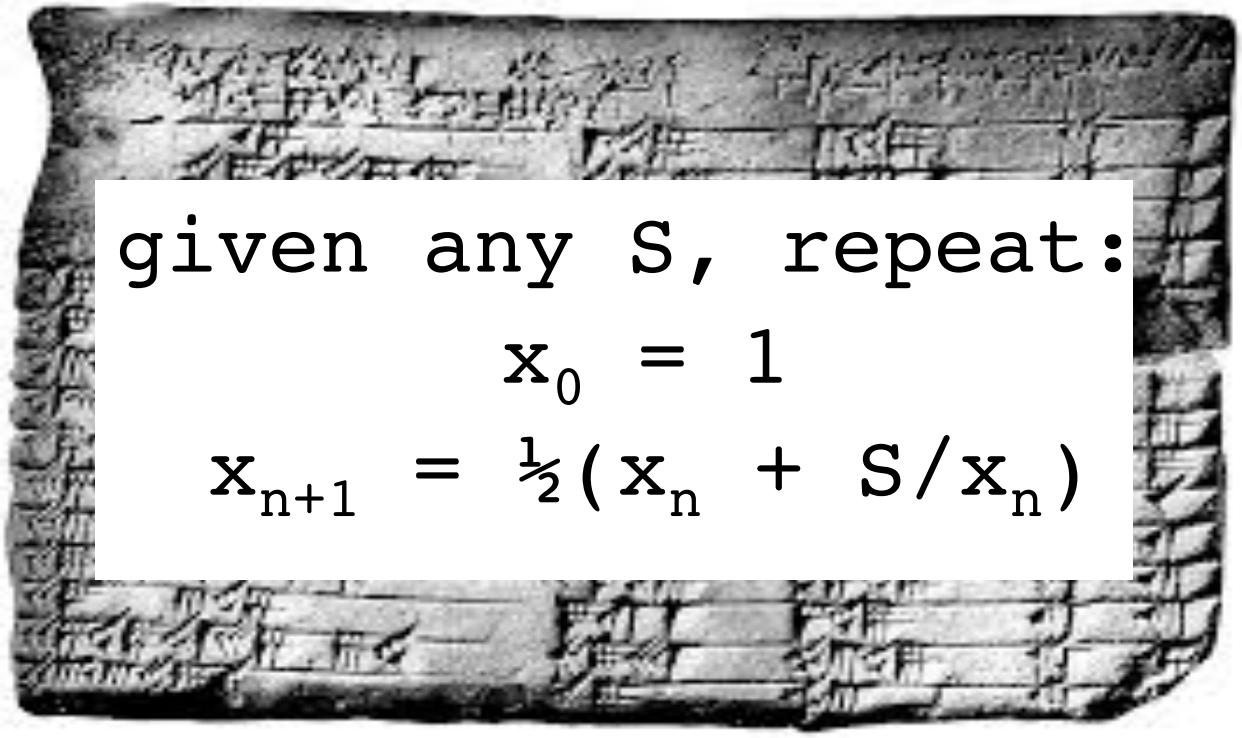
Three Great Ideas in Computing

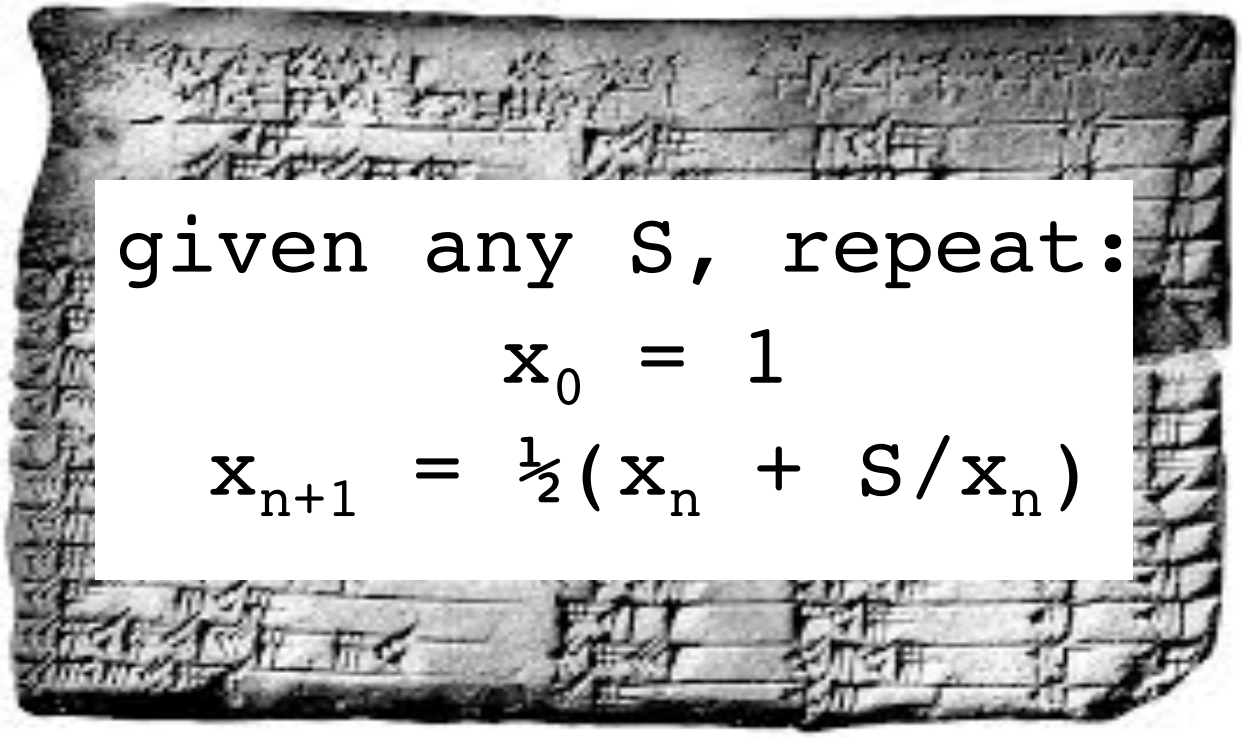
Lev Reyzin

UIC

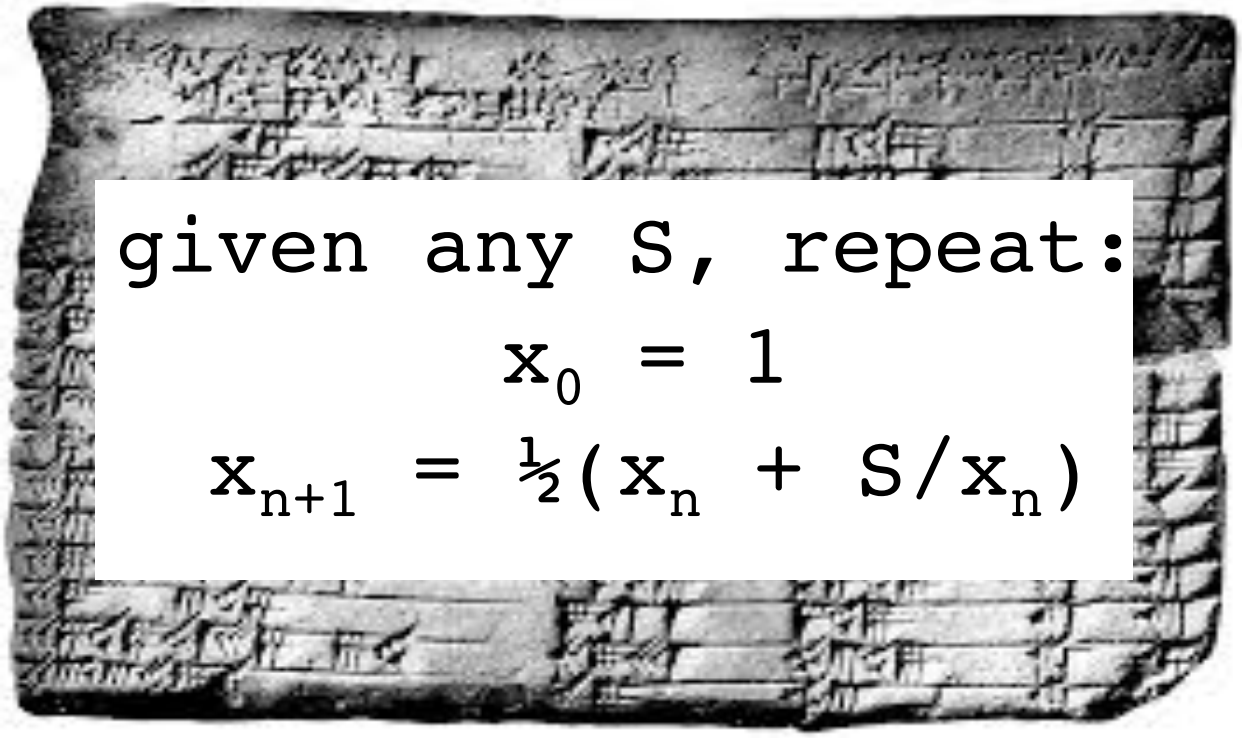
The First Computer Scientists

Babylonians, 1600 B.C.E.


given any S, repeat:
$x_0 = 1$
$x_{n+1} = \frac{1}{2}(x_n + S/x_n)$

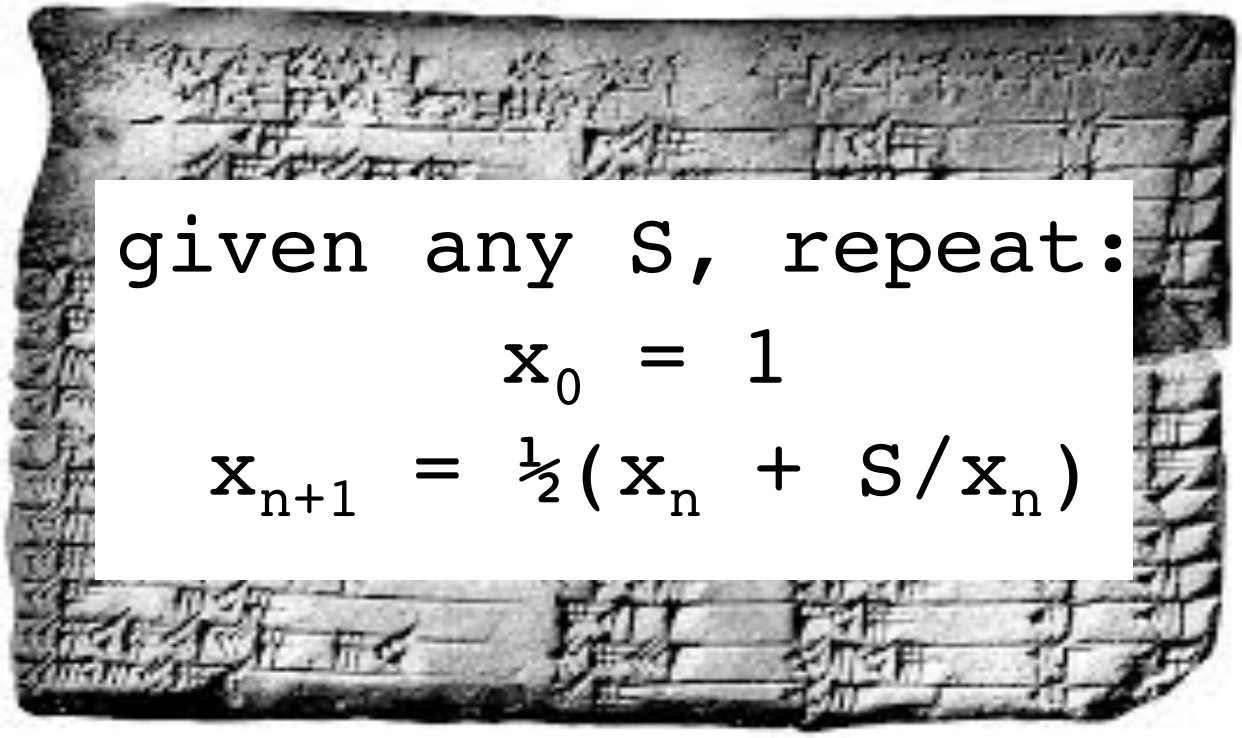


what is $\lim_{x \to \infty} x_n$?



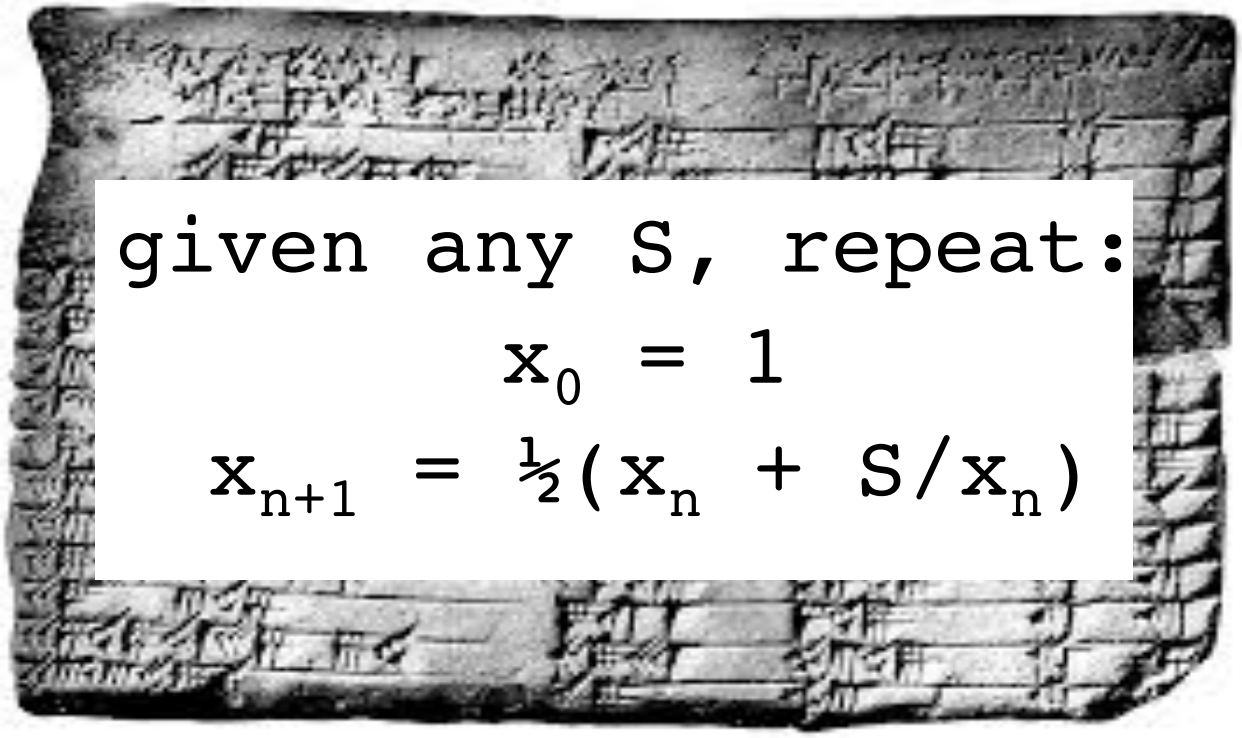
what is $\lim_{x\to\infty} x_n$?

S=10: 1,



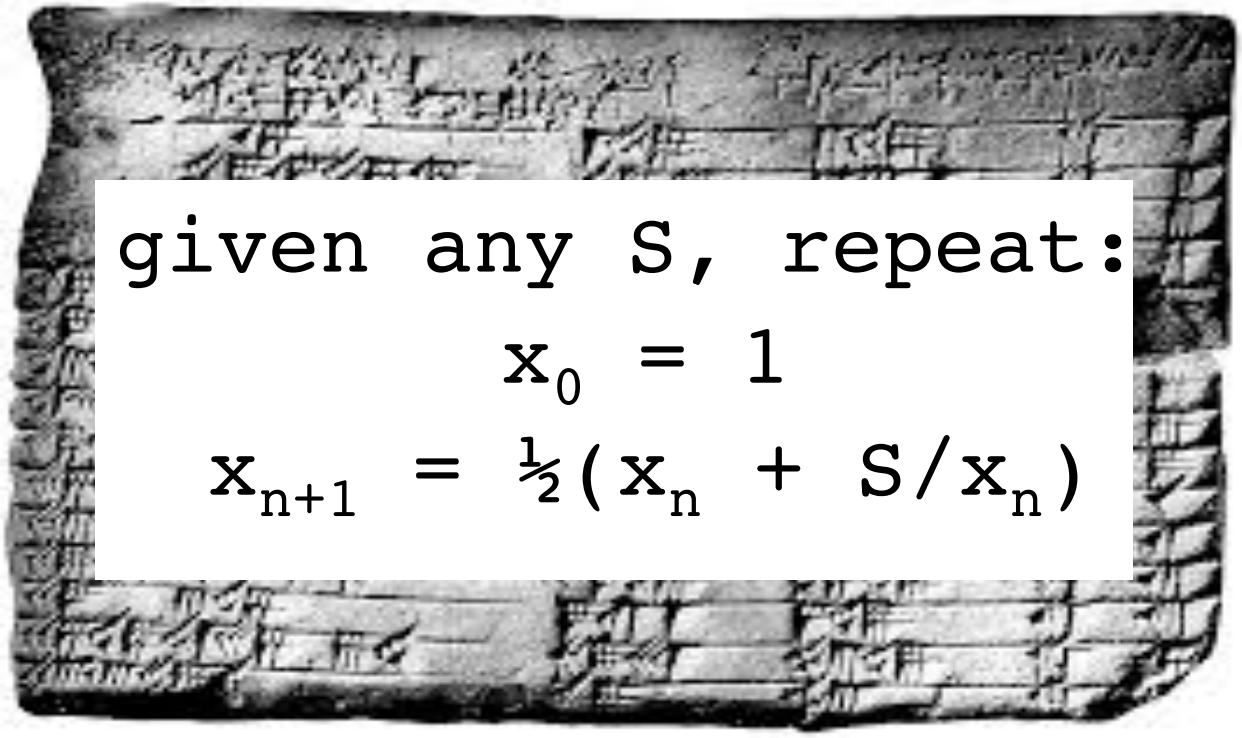
what is $\lim_{x\to\infty} x_n$?

S=10: 1,5.5,



what is $\lim_{x\to\infty} x_n$?

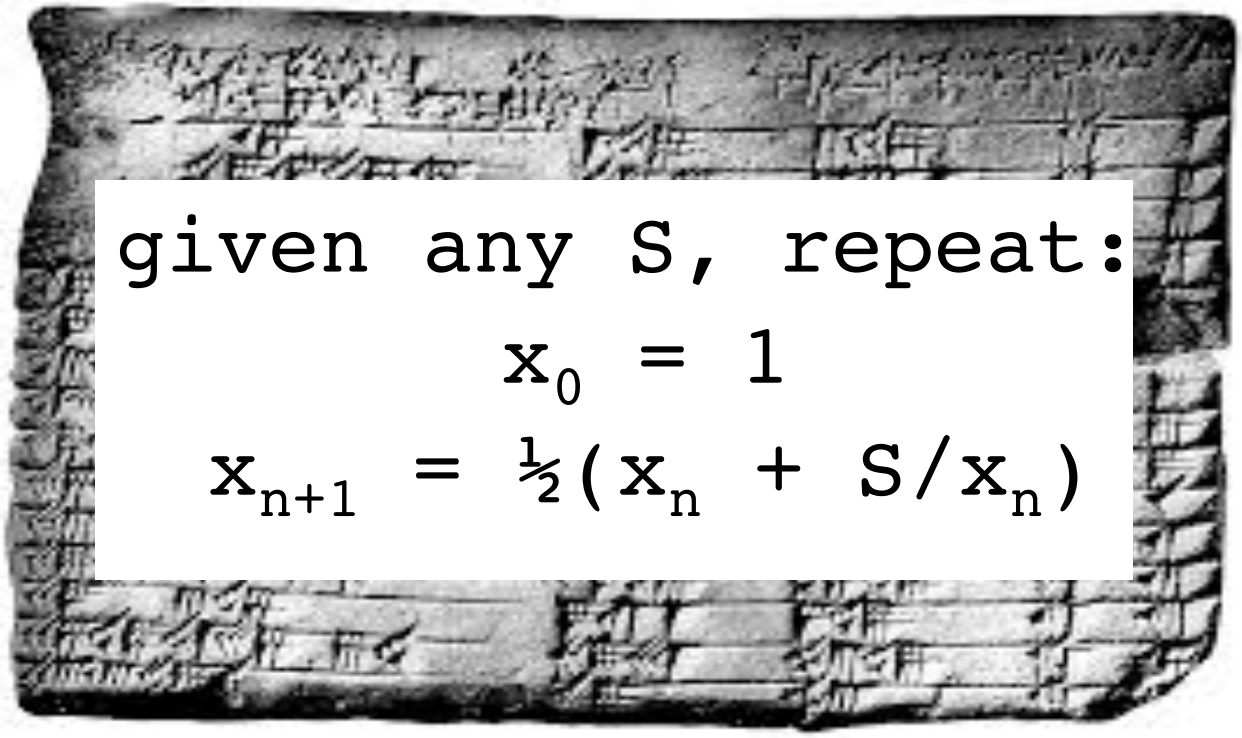
S=10: 1,5.5,3.65,3.20,3.16,…



what is $\lim_{x\to\infty} x_n$?

S=10: 1,5.5,3.65,3.20,3.16,…

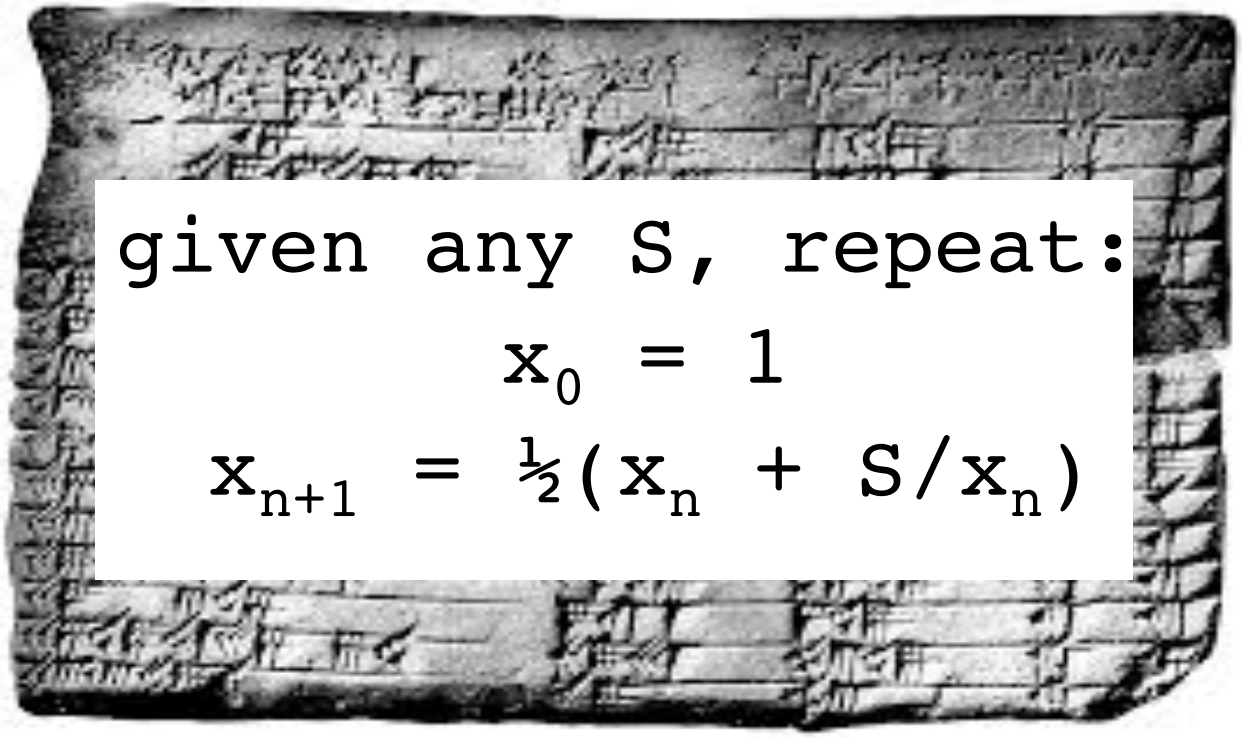
S=100:



what is $\lim_{x\to\infty} x_n$?

S=10: 1,5.5,3.65,3.20,3.16,…

S=100: 1,50.5,26.2,15.0,10.8,10.0,…

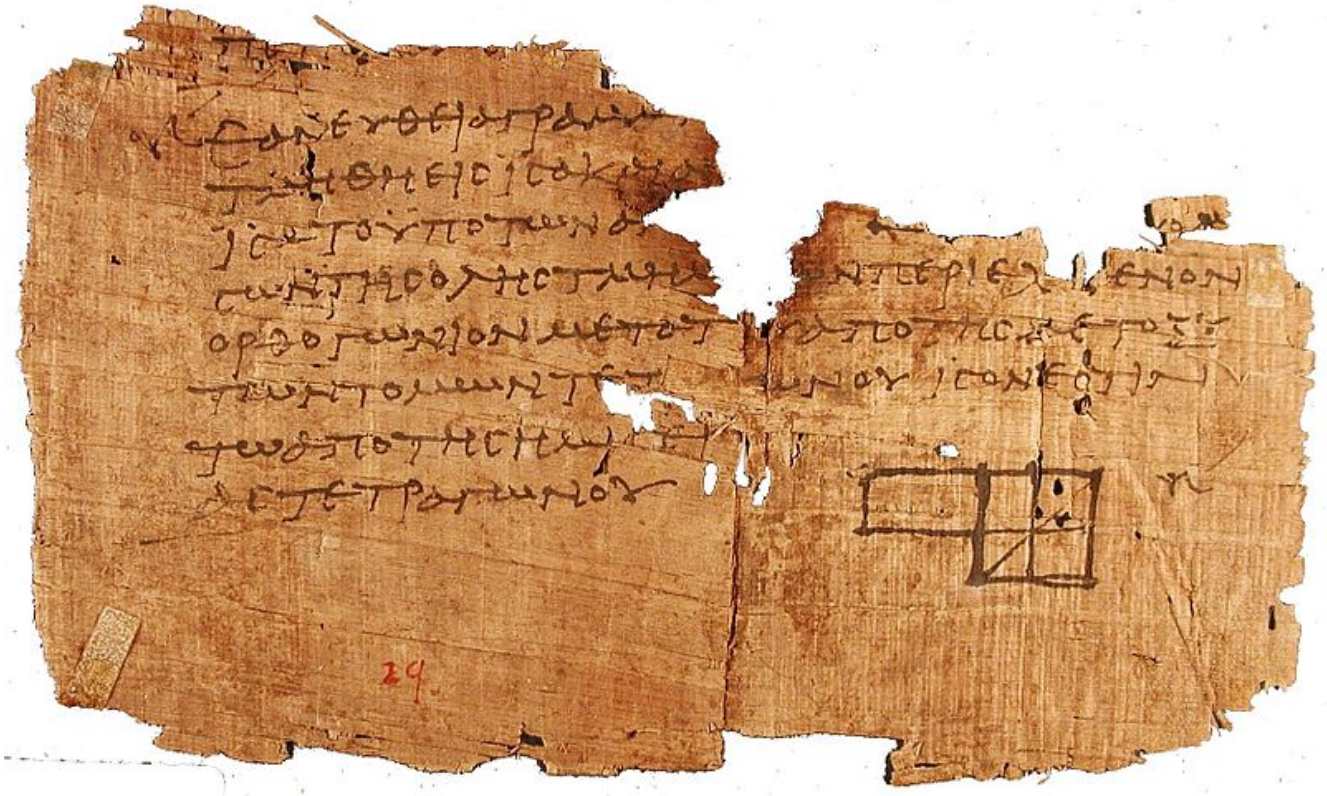


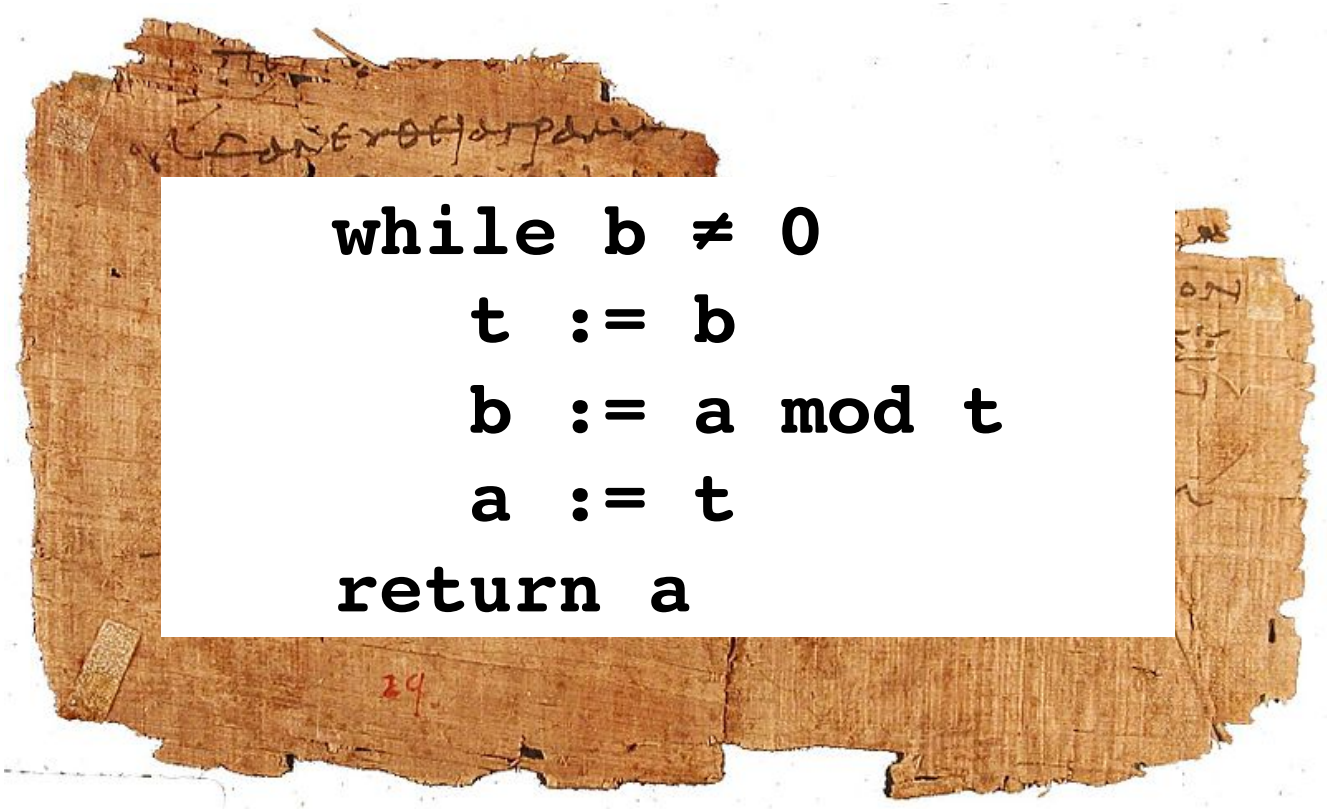
turns out: $\lim_{x\to\infty} x_n = S^{\frac{1}{2}}$

S=10: 1,5.5,3.65,3.20,3.16,…

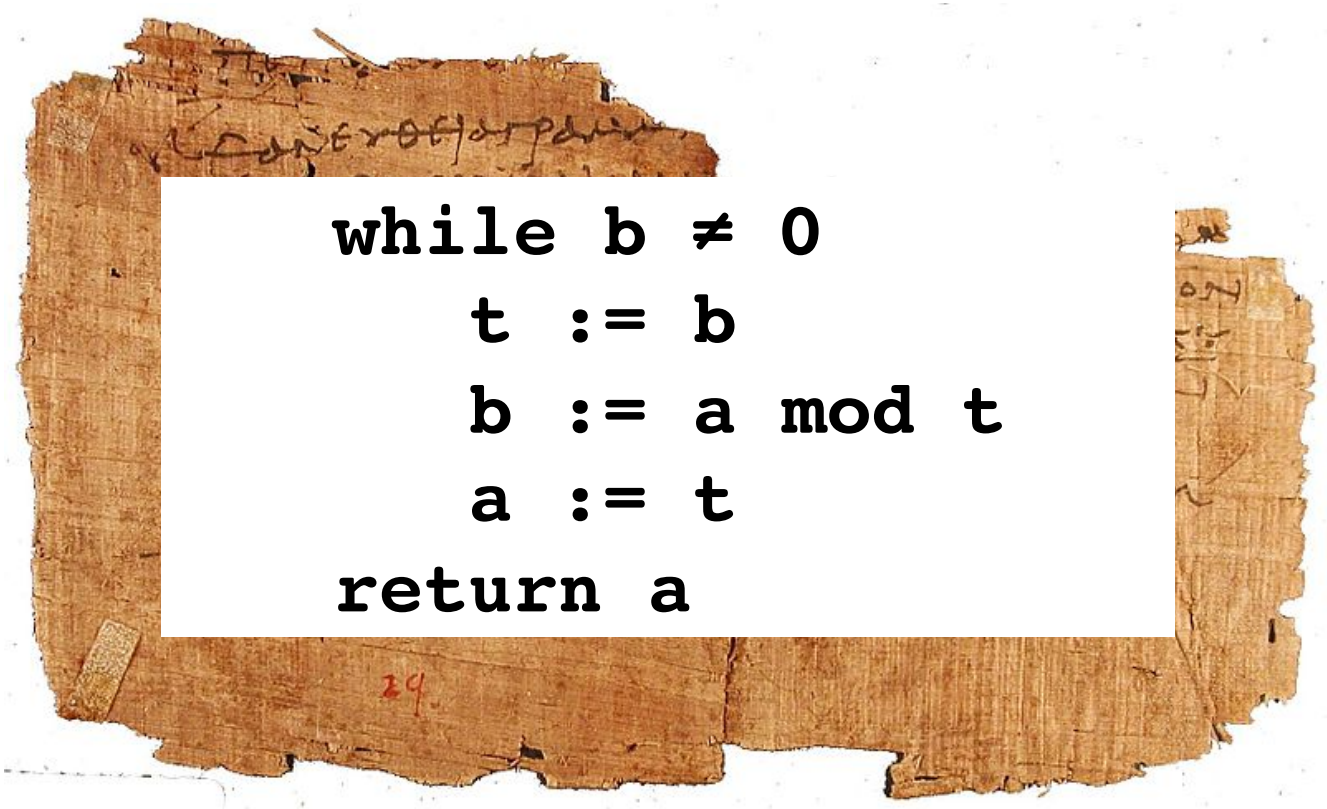
S=100: 1,50.5,26.2,15.0,10.8,10.0,…

Euclid (300 B.C.E.)





```
while b ≠ 0
    t := b
    b := a mod t
    a := t
return a
```



```
while b ≠ 0
    t := b
    b := a mod t
    a := t
return a
```

GCD(12,16)

(16,12)→(12,4)→(4,0)→4

Algorithms Galore!

- Eratosthenes (200 B.C.E.): finding primes
- Liu Hui (263): solving linear equations
- Brahmagupta (628): solving quadratic equations
- Al-Khawarizmi (852): using Arabic numerals

…

- Newton (1671): finding zeros of functions
- Machin (1706): computing pi

…

Hilbert (1928)

Hilbert's Decision Problem

Entscheidungsproblem (decision problem)

"The decision problem is solved if one knows a process which, given a logical expression, permits the determination of its validity… we want to make it clear that for the solution of the decision problem a process would be given by which nonderivability can, in principle, be determined, even though the difficulties of the process would make practical use illusory… the decision problem [is] designated as the main problem of mathematical logic."

Richard Feynman (1918-1988)

Albert Einstein Award (1954)
E. O. Lawrence Award (1962)
Nobel Prize in Physics (1965)
Oersted Medal (1972)
National Medal of Science (1979)

Richard Feynman (1918-1988)

Albert Einstein Award (1954)
E. O. Lawrence Award (1962)
Nobel Prize in Physics (1965)
Oersted Medal (1972)
National Medal of Science (1979)

Feynman's Process

1) Write down the problem.
2) Think real hard.
3) Write down the solution.

(according to Gell-Mann)

Hilbert's Decision Problem

Entscheidungsproblem (decision problem)

"The decision problem is solved if one knows a process which, given a logical expression, permits the determination of its validity… we want to make it clear that for the solution of the decision problem a process would be given by which nonderivability can, in principle, be determined, even though the difficulties of the process would make practical use illusory… the decision problem [is] designated as the main problem of mathematical logic."

Hilbert's ~~Decision~~ Problem

Entscheidungsproblem (decision problem)

"*The decision problem is solved if one knows a process which, given a logical expression, permits the determination of its validity… we want to make it clear that for the solution of the decision problem a process would be given by which nonderivability can, in principle, be determined, even though the difficulties of the process would make practical use illusory… the decision problem [is] designated as the main problem of mathematical logic.*"

Great Idea 1

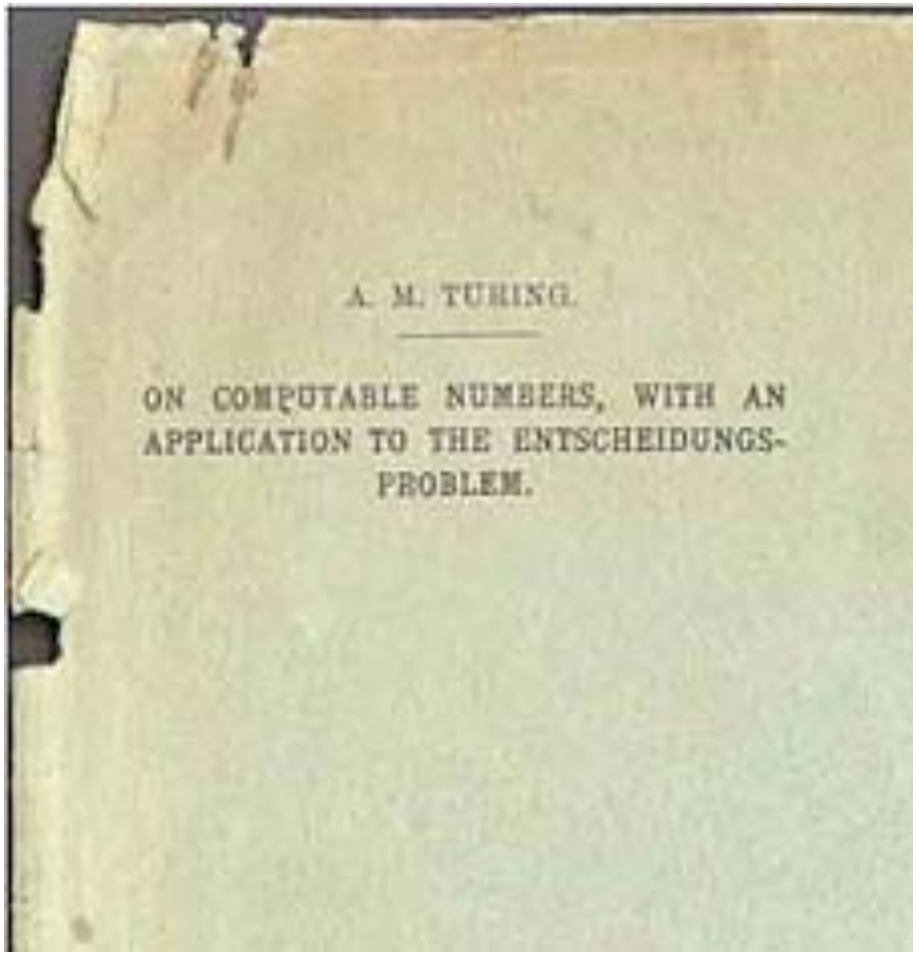
Alan M. Turing (1936)

Alan M. Turing (1936)

Church-Turing thesis: anything that is "effectively computable" is computable by this machine.

Alan M. Turing (1936)

Can just think about our favorite programming language; it's all the same!


A. M. TURING.

ON COMPUTABLE NUMBERS, WITH AN APPLICATION TO THE ENTSCHEIDUNGS-PROBLEM.

Turing's Answer

Hilbert's "process" does not exist for all problems!

Turing's Answer

Hilbert's "process" does not exist for all problems!

Some programs Halt:

```
program_1(x) {
  print("hello");
  for (i=0 to x){
    print(i);
  }
  exit();
}
```

Some programs don't:

```
program_2(x) {
  while(x<x+1) {
    print("nom ");
  }
  print("goodbye");
  exit();
}
```

The Halting Problem

Given a program P, how can we tell if it halts on x?

```
P(x) {
   int y = 40;
   while(x > y || y < 100) {
      for(int z = 1; z < x+y; x++) {
         y = y + 2;
         …
      }
      …
   }
}
```

The Halting Problem

Imagine there exists H(P,x) that given a program P and input x, returns 1 if P(x) halts and 0 if P(x) loops forever.

if such an H exists, we can create these two programs:

```
stops_on_self(P){
  return H(P,P);
}
```

```
garblygook(P) {
  if (stops_on_self(P))
    while(true) {};
  else
    exit();
}
```

The Halting Problem

Imagine there exists H(P,x) that given a program P and input x, returns 1 if P(x) halts and 0 if P(x) loops forever.

if such an H exists, we can create these two programs:

```
stops_on_self(P){
  return H(P,P);
}
```

```
garblygook(P) {
  if (stops_on_self(P))
    while(true) {};
  else
    exit();
}
```

What happens when you run `garblygook(garblygook)`?

The Halting Problem

Imagine there exists H(P,x) that given a program P and input x, returns 1 if P(x) halts and 0 if P(x) loops forever.

if such an H exists, we can create these two programs:

```
stops_on_self(P){
  return H(P,P);
}
```

```
garblygook(P) {
  if (stops_on_self(P))
    while(true) {};
  else
    exit();
}
```

What happens when you run `garblygook(garblygook)`?

The Halting Problem

Imagine there exists H(P,x) that given a program P and input x, returns 1 if P(x) halts and 0 if P(x) loops forever.

if such an H exists, we can create these two programs:

```
stops_on_self(P){
  return H(P,P);
}
```

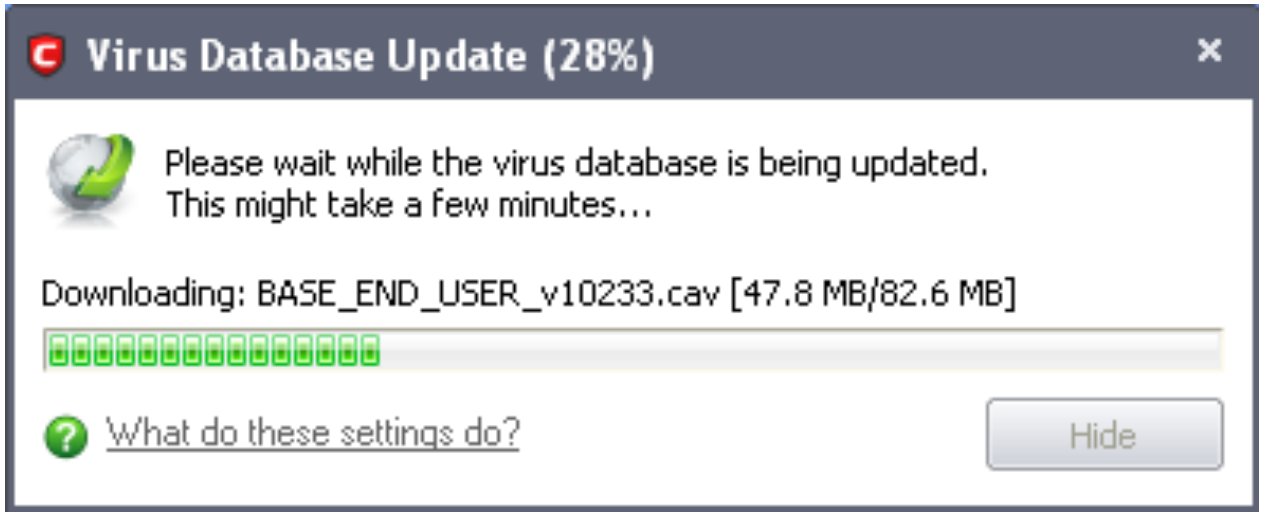
```
garblygook(P) {
  if (stops_on_self(P))
    while(true) {};
  else
    exit();
}
```

If `(garblygook(garblygook))` halts, then it loops forever. If it loops forever, it halts. Contradiction!

So, H cannot exist. Called undecidability!

So What?

Tons of real-life applications: e.g. computer viruses.



Lots of Open Problems!

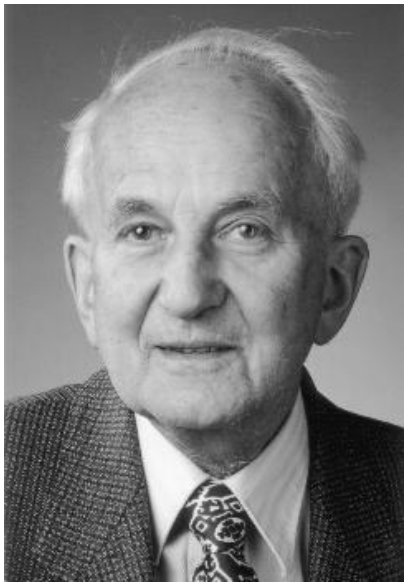
2x2 matrix mortality: given a finite list of 2x2 matrixes, find a sequence that multiplies to 0.

$$\begin{pmatrix} 2 & 4 \\ -1 & 3 \end{pmatrix}, \begin{pmatrix} 0 & 1 \\ 4 & -2 \end{pmatrix}, \begin{pmatrix} 3 & -4 \\ -1 & 0 \end{pmatrix}$$

A B C

We don't even know if this is decidable!

Collatz Conjecture (1937)



"Take any natural number n. If n is even, divide it by 2. If n is odd, multiply it by 3 and add 1. Repeat the process indefinitely. Conjecture: no matter what number you start with, you will eventually reach 1."

Collatz Conjecture (1937)

1✓, 2→1✓, 3→10→5→16→8→4→2→1✓,
4→2→1✓, 5→16→8→4→2→1✓,
6→10→5→16→8→4→2→1✓, 7→22→11→
34→17→52→26→13→40→20→10→5→
16→8→4→2→1✓, 8→4→2→1✓

"Take any natural number n. If n is even, divide it by 2. If n is odd, multiply it by 3 and add 1. Repeat the process indefinitely. Conjecture: no matter what number you start with, you will eventually reach 1."

Collatz Conjecture (1937)

```
Collatz(n) {
  while(n > 1) {
    if (n % 2 == 0) n = n/2;
    else n = 3*n+1;
  }
}
```

Conjecture: `Collatz(n)` Halts for all n.

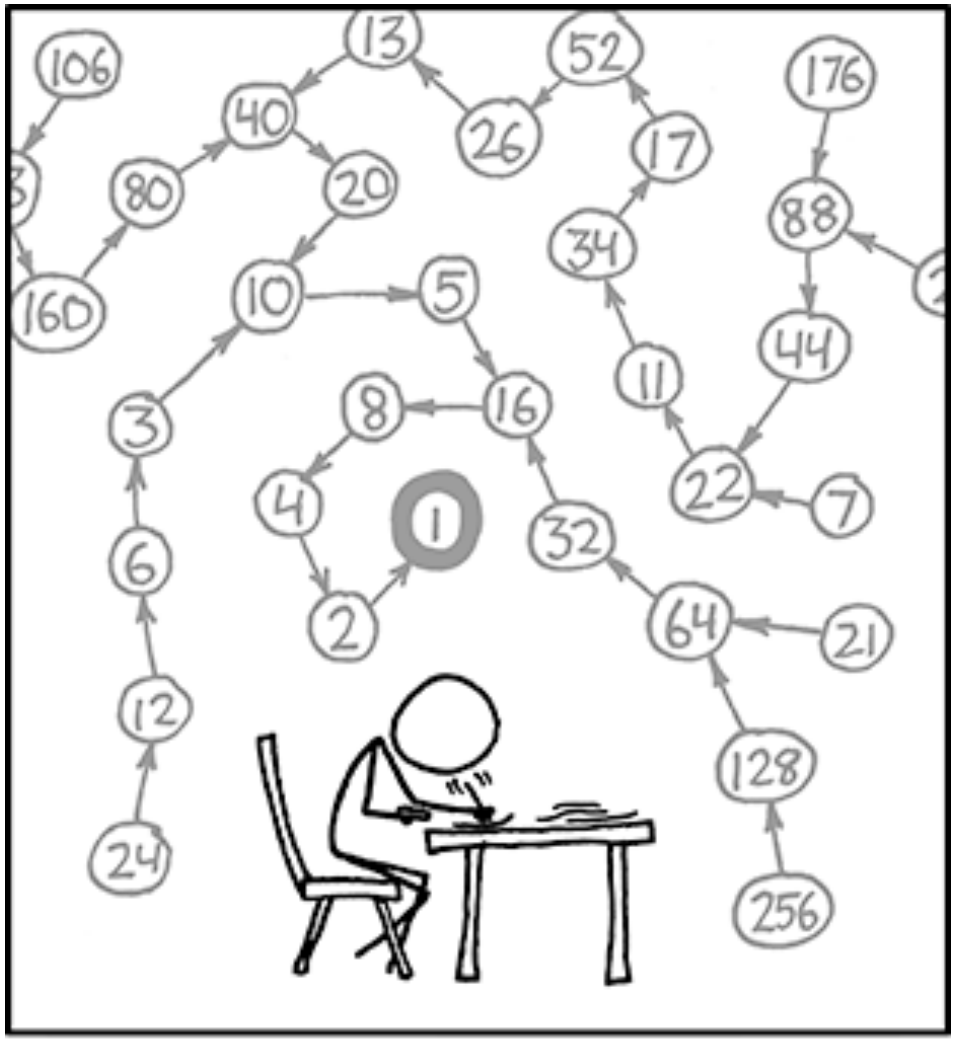
"Take any natural number n. If n is even, divide it by 2. If n is odd, multiply it by 3 and add 1. Repeat the process indefinitely. Conjecture: no matter what number you start with, you will eventually reach 1."

Collatz Conjecture (1937)

```
Collatz(n) {
  while(n > 1) {
    if (n % 2 == 0) n = n/2;
    else n = 3*n+1;
  }
}
```

Conjecture: `Collatz(n)` Halts for all n.

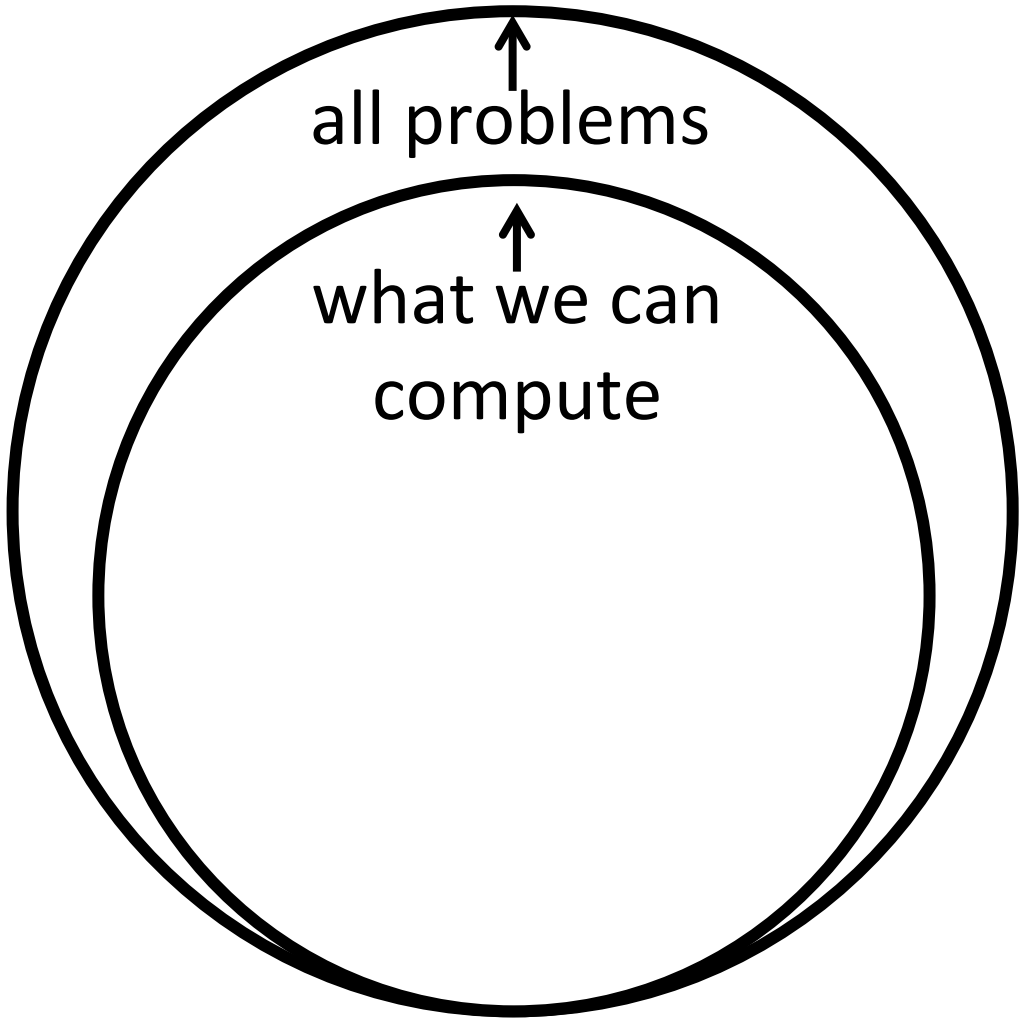
Because the Collatz conjecture has a "yes/no" answer, it technically cannot be "undecidable," but John Conway showed it's very close to an uncomputable problem.



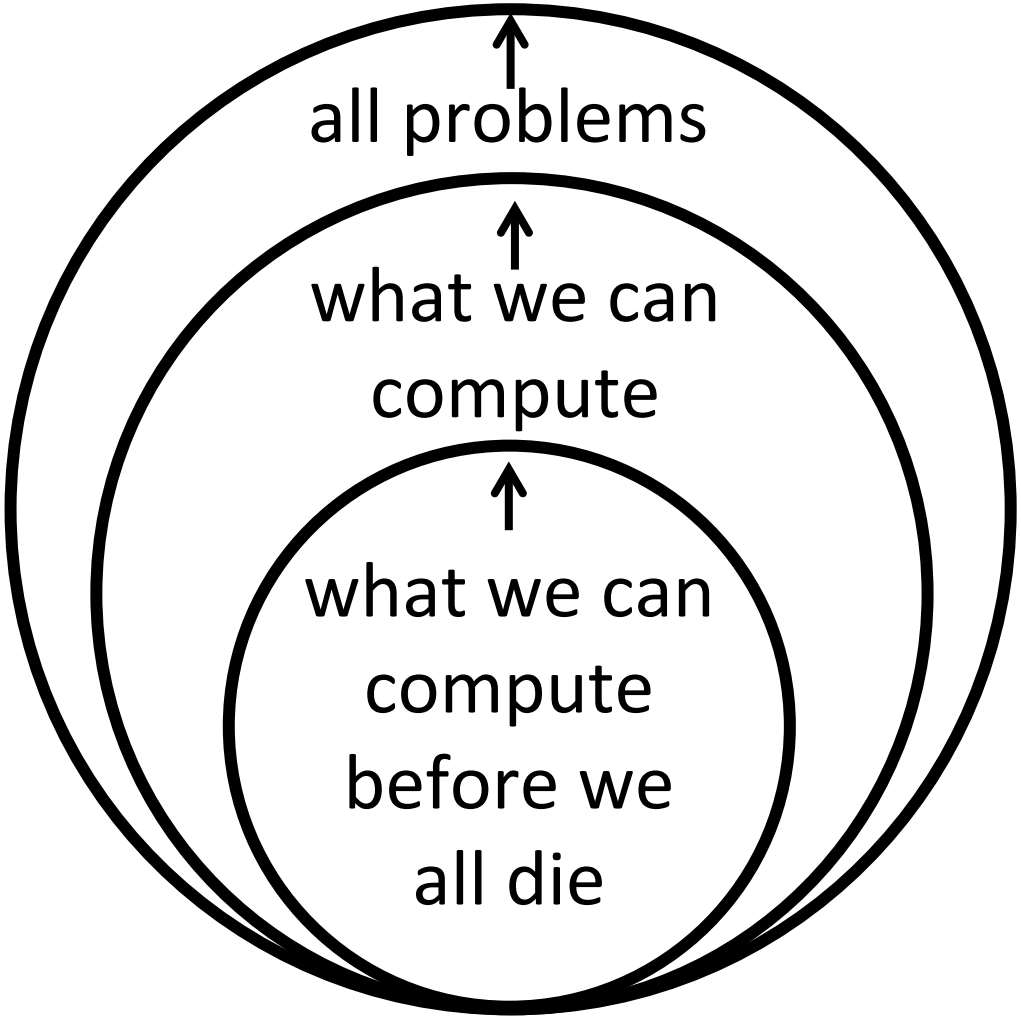
THE COLLATZ CONJECTURE STATES THAT IF YOU PICK A NUMBER, AND IF IT'S EVEN DIVIDE IT BY TWO AND IF IT'S ODD MULTIPLY IT BY THREE AND ADD ONE, AND YOU REPEAT THIS PROCEDURE LONG ENOUGH, EVENTUALLY YOUR FRIENDS WILL STOP CALLING TO SEE IF YOU WANT TO HANG OUT.

Great Idea 2

Computability Theory



Complexity Theory



What is the Shortest Route Visiting all Major US Cities?

Traveling Salesperson Problem

Given a map and a mileage limit, the problem is to determine if you can visit all cities on the map without exceeding the mileage limit?

Traveling Salesperson Problem

Given a map and a mileage limit, the problem is to determine if you can visit all cities on the map without exceeding the mileage limit?

Clearly decidable – we can try all the possibilities!

But would we live long enough to find out the answer?

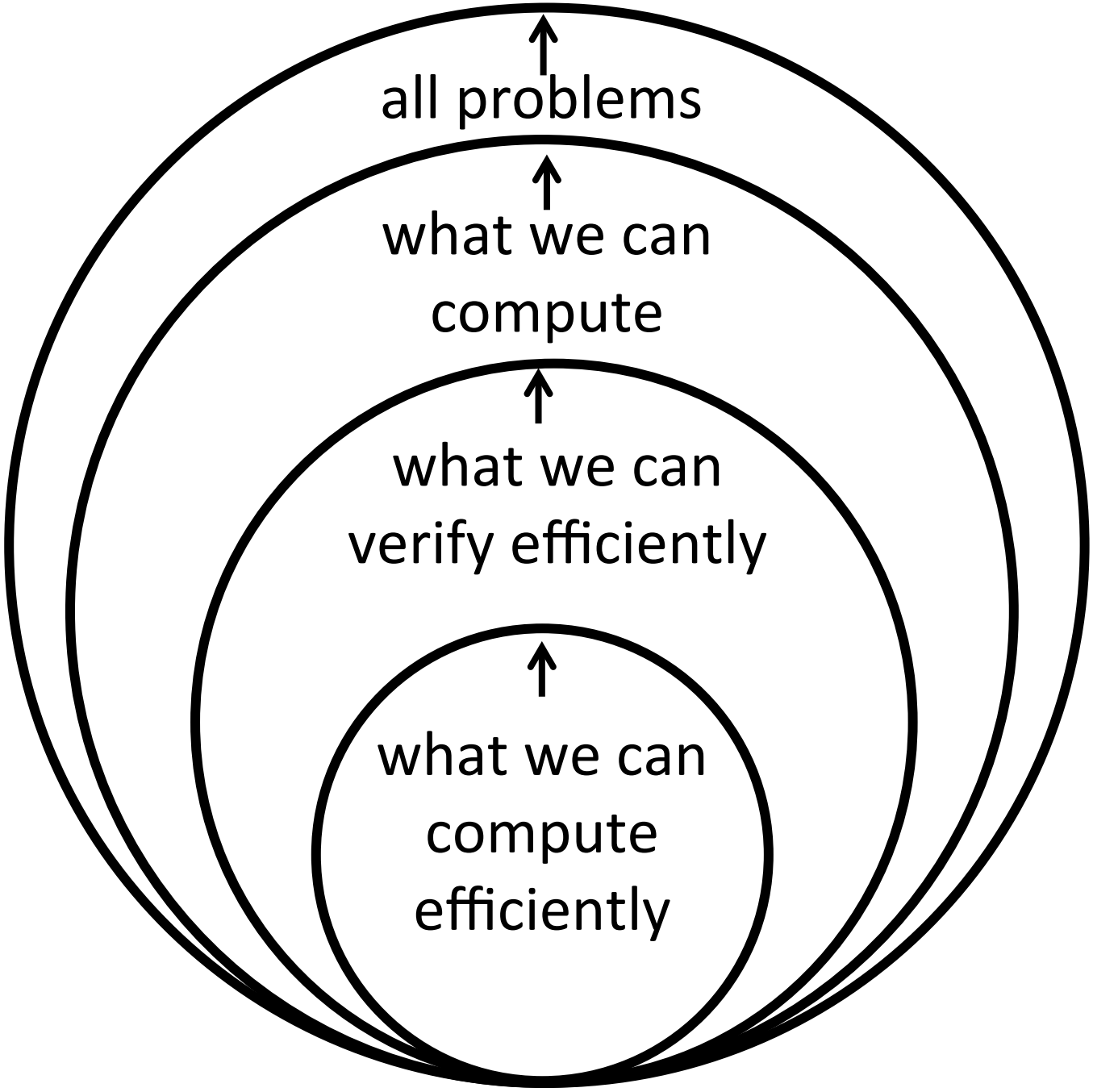
Traveling Salesperson Problem

Given a map and a mileage limit, the problem is to determine if you can visit all cities on the map without exceeding the mileage limit?

For 60 cities this would be 60! $> 10^{81}$, which happens to be 10 times the estimated number of atoms in the universe.

P vs NP

- Some problems are easy to check (NP)
- Some problems are easy to solve (P)
- TSP is in NP but we ~~think~~ are very confident that it is not P. In fact it's one of many "hardest" problems in NP.
- When we see one of these "hardest" problems, we cannot hope to solve it exactly.


all problems
what we can compute
what we can verify efficiently
what we can compute efficiently

Using Hardness

Imagine Alice, Bob, and Eve all walk into a room. They have never met before. Everything they say to each other is heard by all 3. Can Alice and Bob exchange a secret that Eve doesn't know?

Alice

Eve

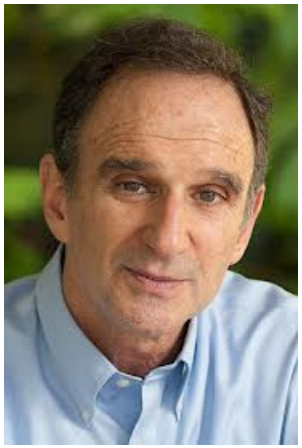
Bob

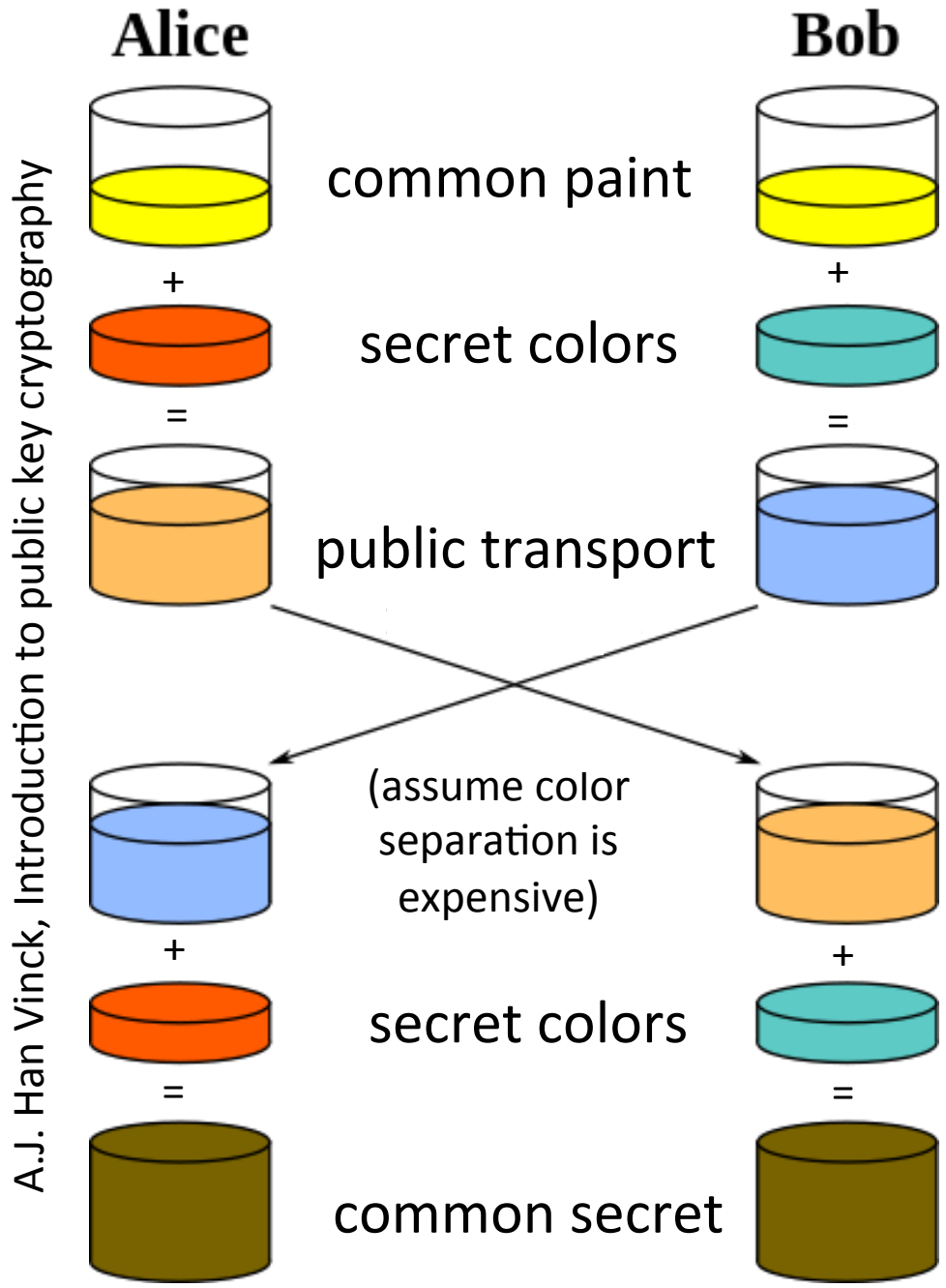
Using Hardness

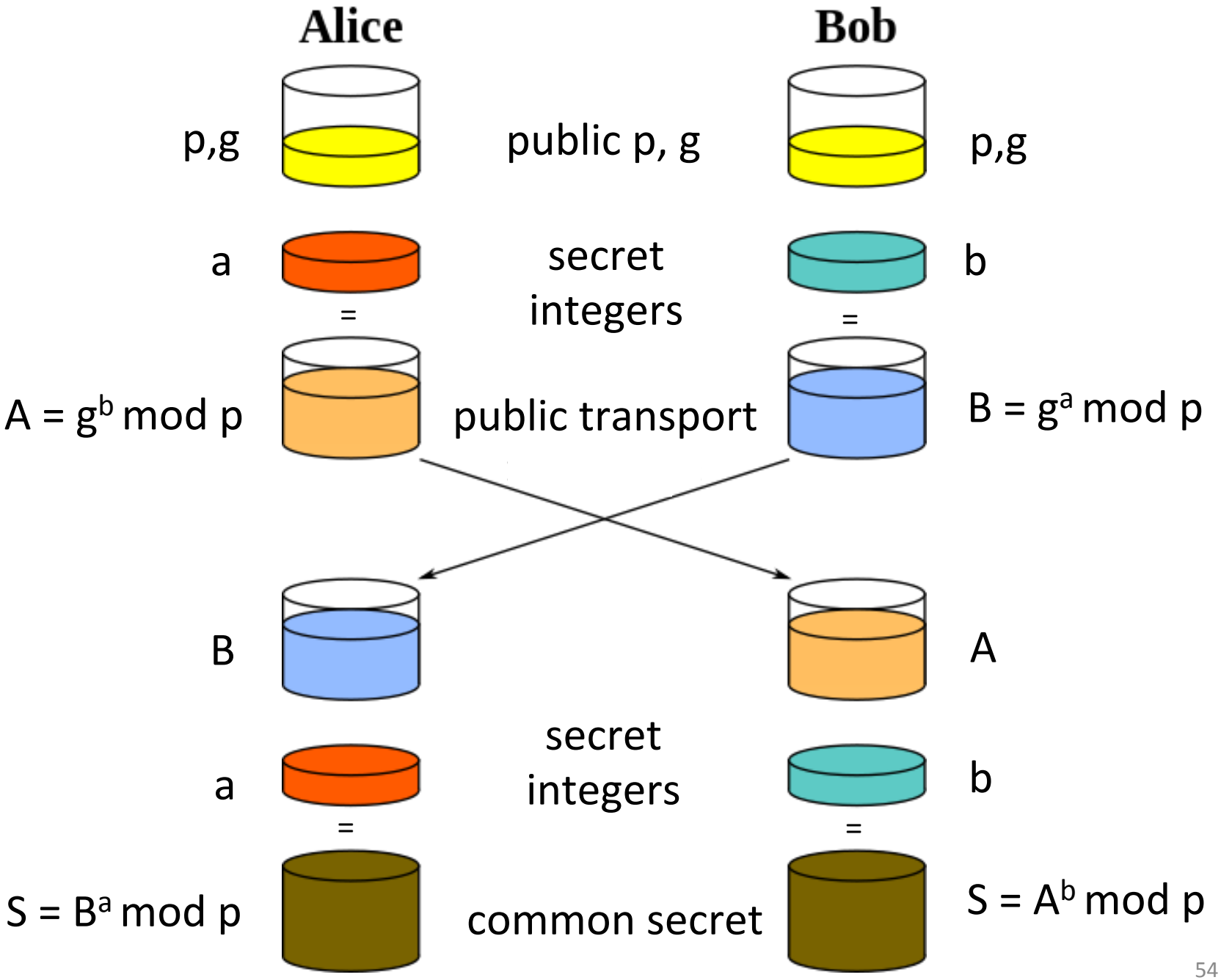
Imagine Alice, Bob, and Eve all walk into a room. They have never met before. Everything they say to each other is heard by all 3. Can Alice and Bob exchange a secret that Eve doesn't know?

Shockingly, Yes!

[Diffie-Hellman '76]




Alice
Bob
common paint
+
secret colors
=
public transport
(assume color separation is expensive)
+
secret colors
=
common secret
A.J. Han Vinck, Introduction to public key cryptography


Alice
Bob
p,g
public p, g
p,g
a
secret integers
b
=
=
A = g^b mod p
public transport
B = g^a mod p
B
A
a
secret integers
b
=
=
S = B^a mod p
common secret
S = A^b mod p

Using Hardness

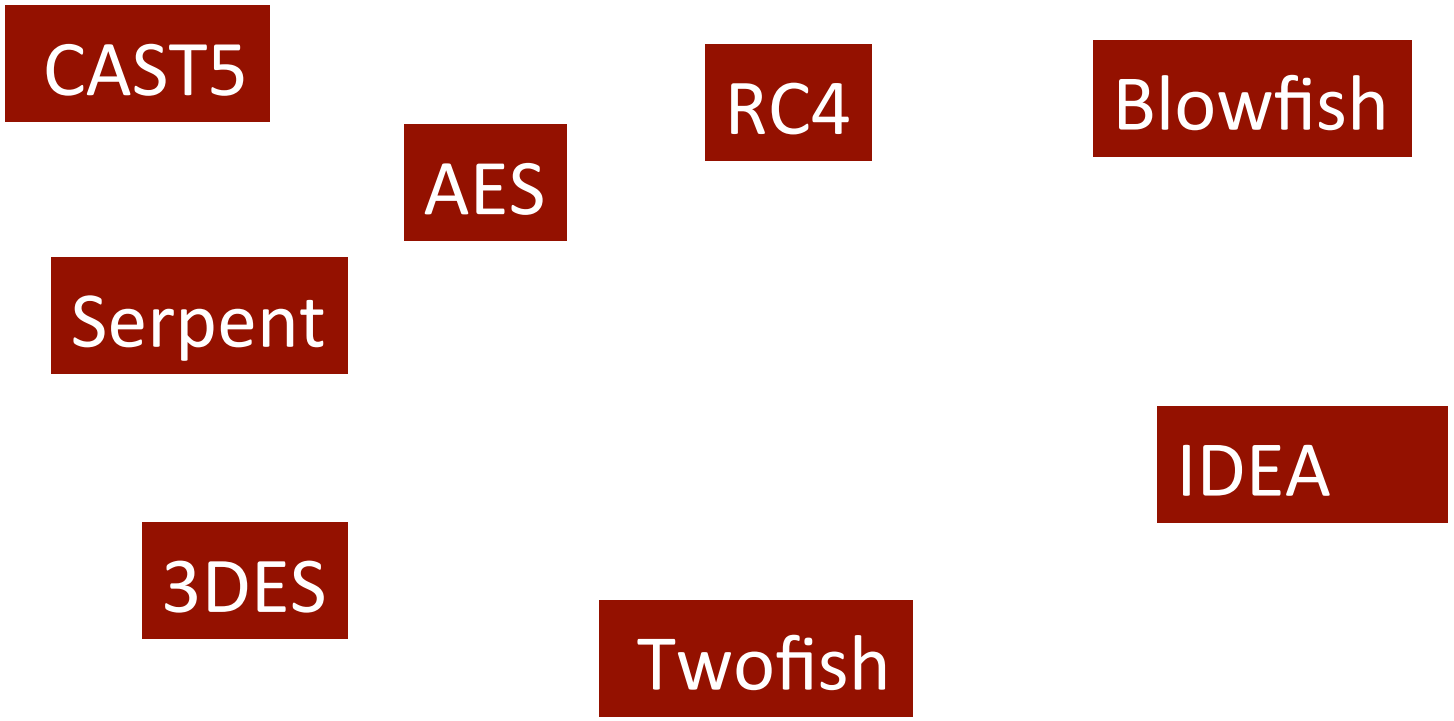
Diffie-Hellman assumption (variant):

given
an integer g
a prime p,
g^x mod p and g^y mod p,
finding g^{xy} mod p cannot
be done efficiently

If the Diffie-Hellman assumption true,
Eve cannot know the common secret without doing an exponential amount of computation.

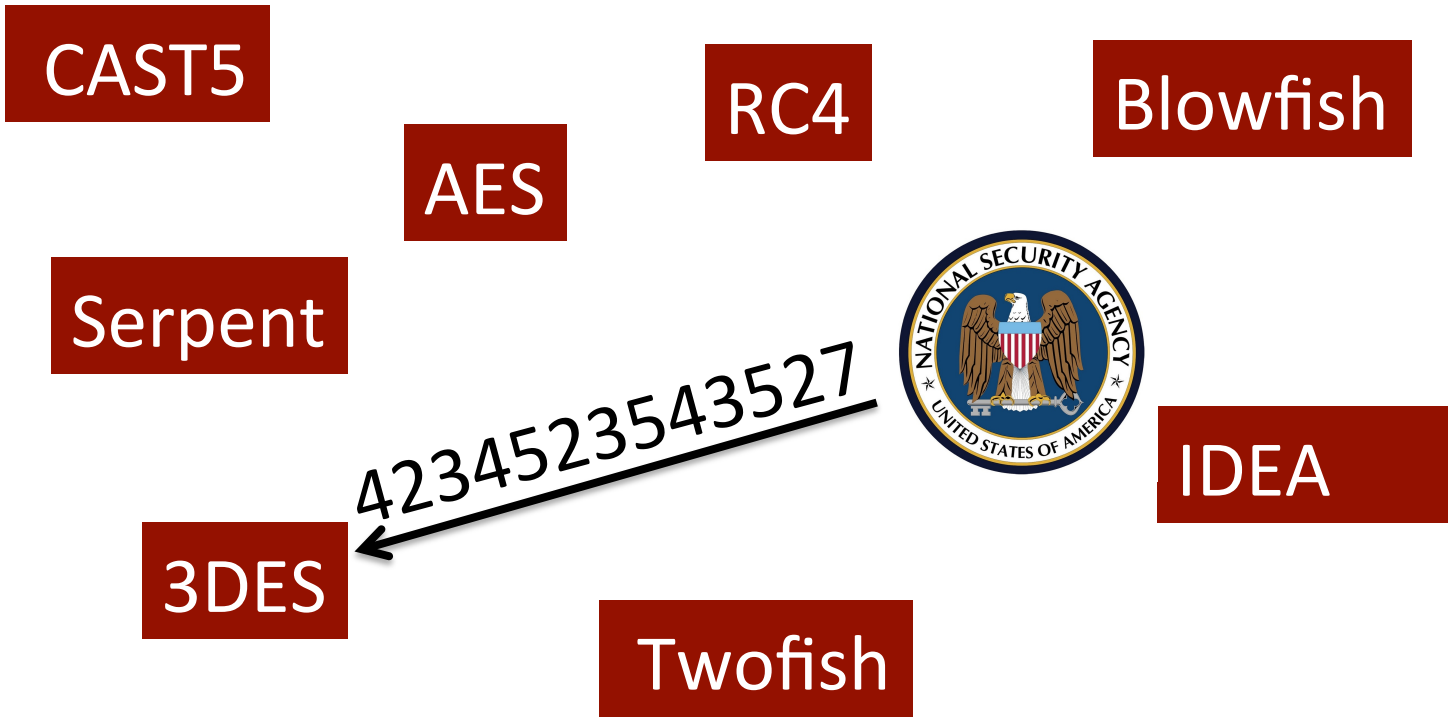
Using the Shared Secret

Alice and Bob can then use their shared secret value for symmetric cryptography.

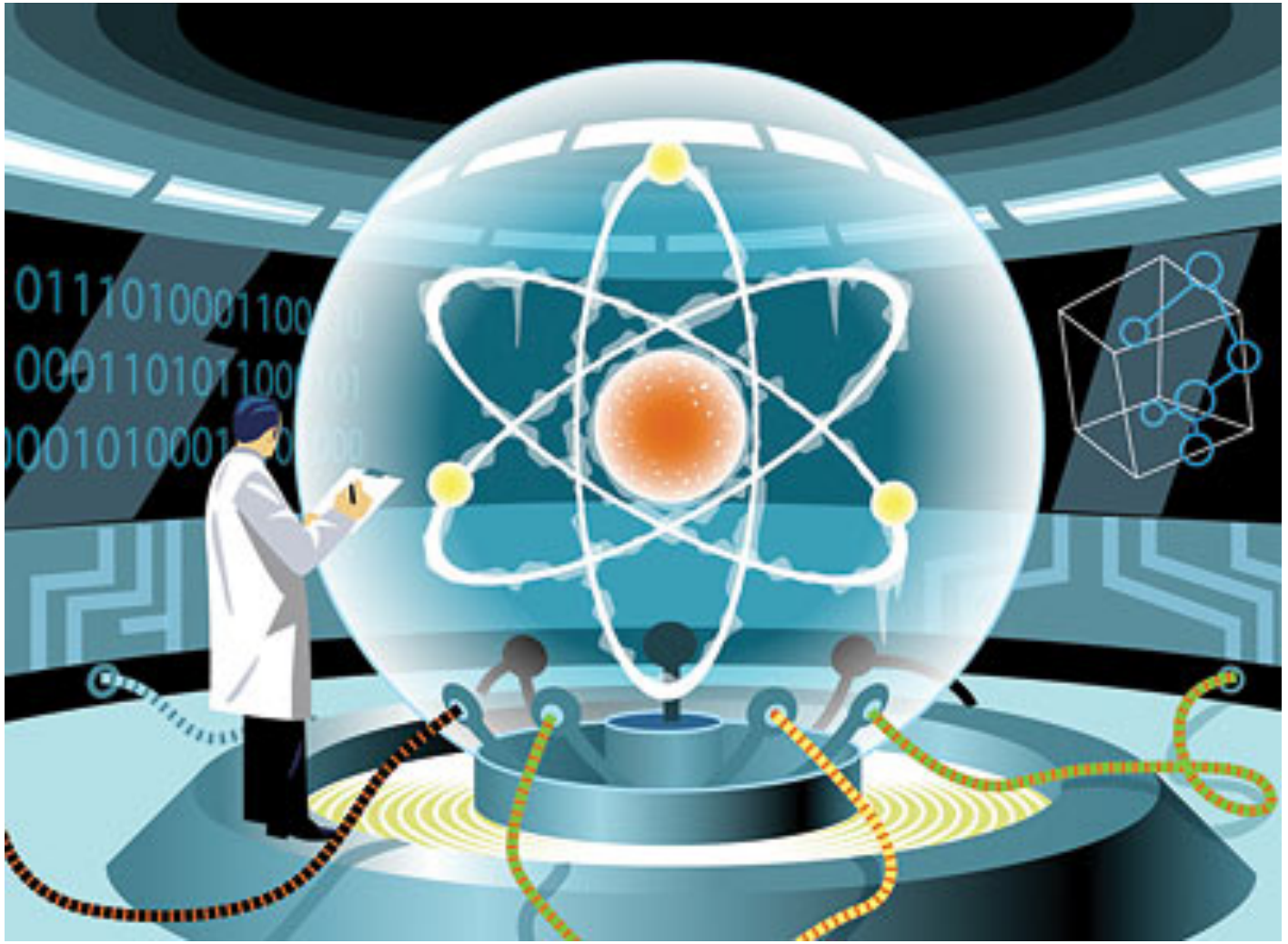


Using the Shared Secret

Alice and Bob can then use their shared secret value for symmetric cryptography.



Quantum Computers



Quantum Computers

Factoring and “discrete log” are easy on a quantum computer. [Shor ’97]

Breaks the Diffie-Hellman assumption, among other things.

How do we test the security of a crypto system?

How do we do efficient multi-party computation?

What hard problems are resistant to quantum computers?

Can we implement practical homomorphic encryption?

Crypto in One Slide

Hello, I have a message of great consequence. It is very important that ….

0101010111000010110110
1000101011001010100010
0100011011000011110101
1001100101010110110010
1001110001101101…

Hello, I have a message of great consequence. It is very important that ….

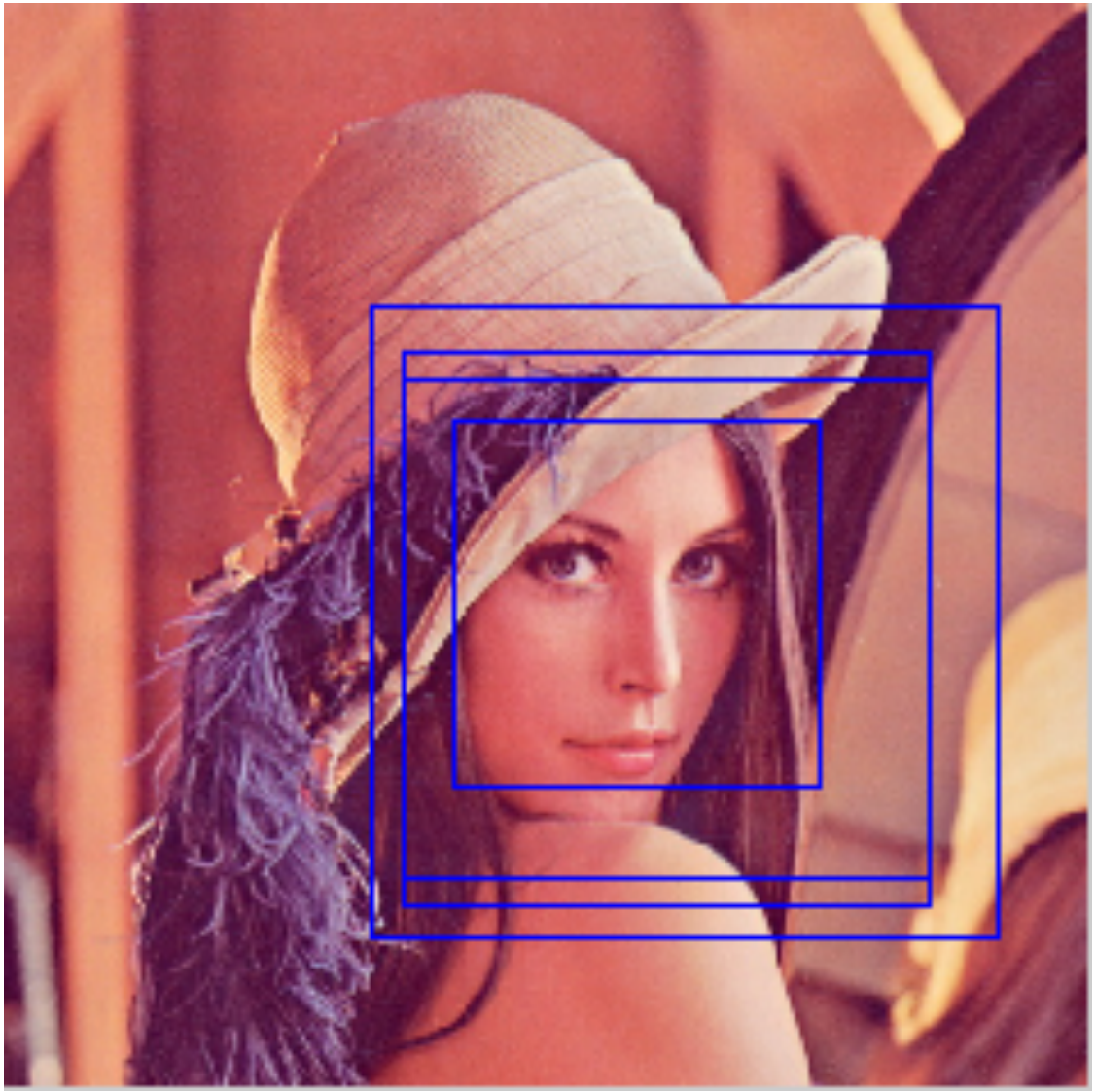
Breaking Crypto = Hard?

Hello, I have a message of great consequence. It is very important that ….

Hello, I have a message of great consequence. It is very important that ….

0101010111000010110110
1000101011001010100010
0100011011000011110101
1001100101010110110010
1001110001101101…

Many Real World Problems are Easy: Face Recognition



Viola-Jones [’01]

Great Idea 3

How Would You Program a Computer to Detect Faces?

- Think really hard about what defines a face and program it in?

How Would You Program a Computer to Detect Faces?

- Think really hard about what defines a face and program it in?

- No! Give the computer examples and let it learn.

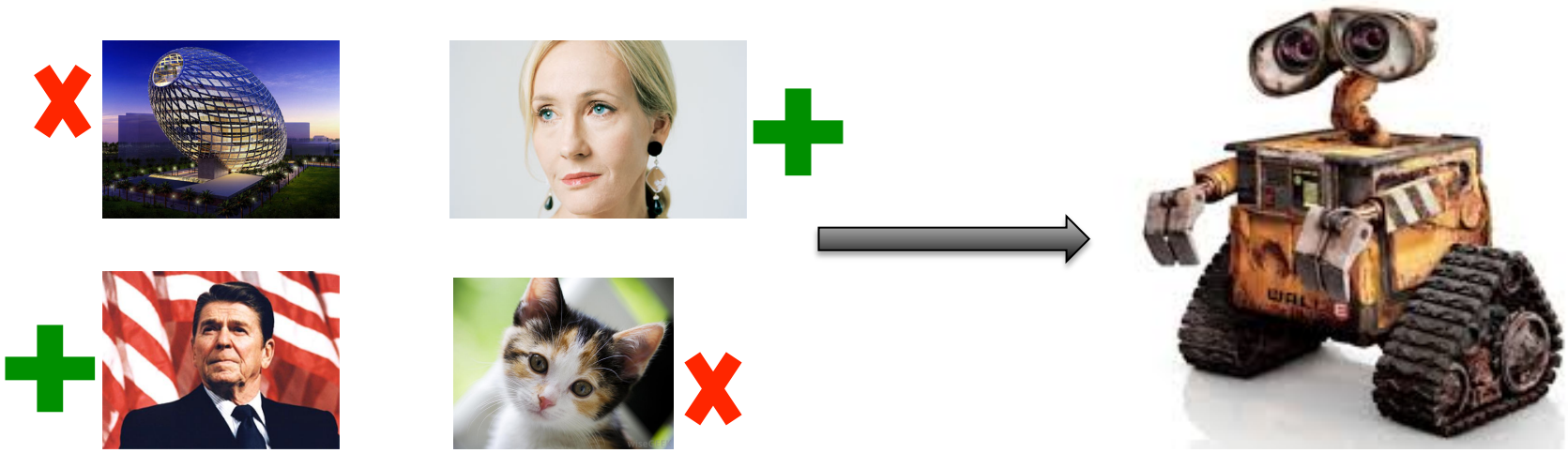
A Theoretical Framework

Probably Approximately Correct Learning: A function family is learnable if for any distribution over the data, the "target" can be approximated with high probability by giving a computer not too many samples from the distribution.

Valiant ['84]
"*A theory of the learnable*"

A Theoretical Framework

Probably Approximately Correct Learning: A function family is learnable if for any distribution over the data, the "target" can be approximated with high probability by giving a computer not too many samples from the distribution.



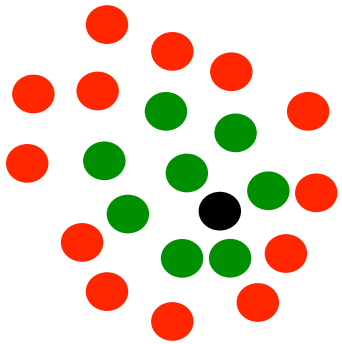
Huh?

- But isn’t making a computer “learn” even harder?

Huh?

- But isn't making a computer "learn" even harder?

- Not really! Imagine something like "nearest neighbor"



Face Detection




nearest neighbor isn’t so good

Surprising Results

Boosting: To get a probably approximately correct learner, one just needs, with some small probability, to be able to do better than random guessing on every distribution – that's it!

Schapire ['90]
"The strength of weak learnability"

Example: Spam Detection

- Hard to program. Easy to come up with rules.
  - Viagra -> spam
  - homework -> not spam
  - Integral -> not spam
  - $$$-> spam

- This idea, and others, was used in face-detection by Viola-Jones [’01]!

Support Vector Machines

Linear functions are easy to learn.

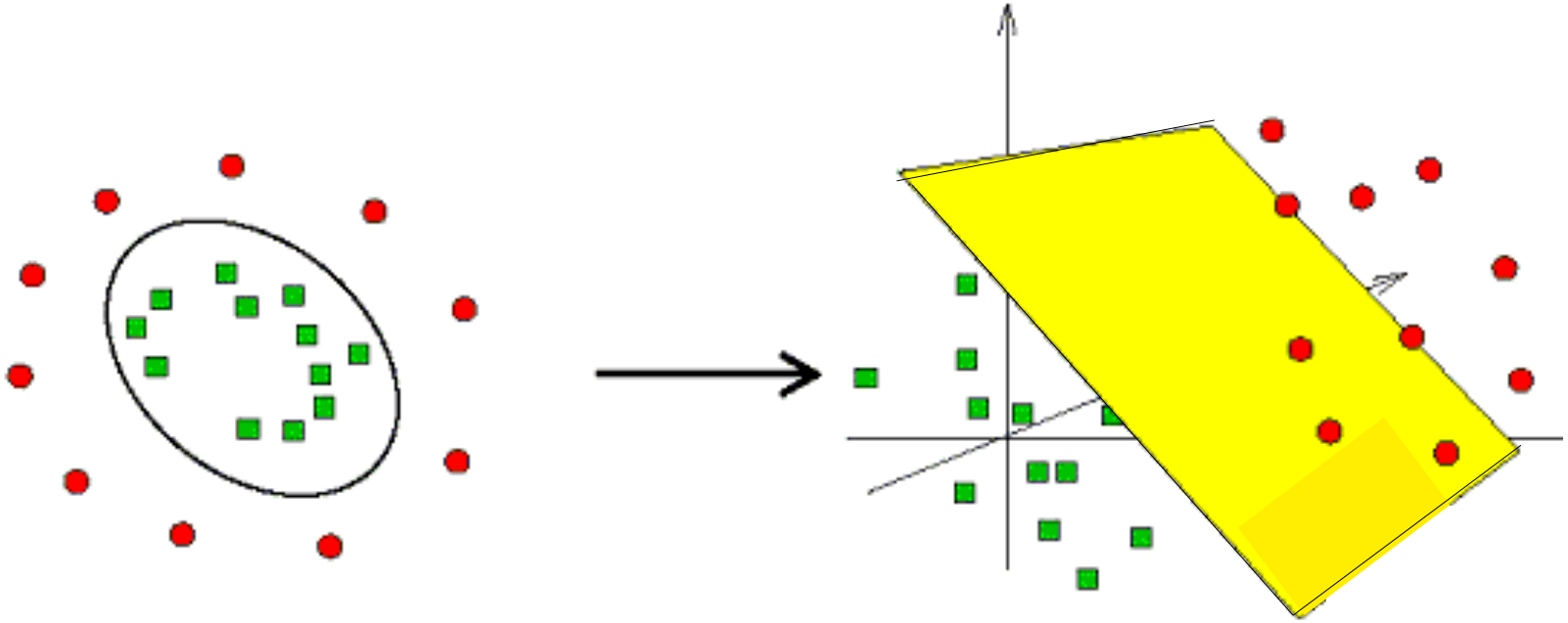
Put the data in high dimension to make it linearly separable!

Cortes and Vapnik [’95]

Support Vector Machines

Linear functions are easy to learn.

Put the data in high dimension to make it linearly separable!



complex in low dimensions

simple in higher dimensions

Many Simple Things We Don't Know

- Are decision trees learnable?
- Is learning hard? (We think so, but no so easy to prove.)
- How much harder is it to learn when there is noise? (Ties into cryptography.)

Many Other Beautiful Great Ideas

Zero Knowledge Proofs

Let’s say Alice proves the Riemann Hypothesis. Can she convince Bob her proof is correct without revealing anything about her proof?

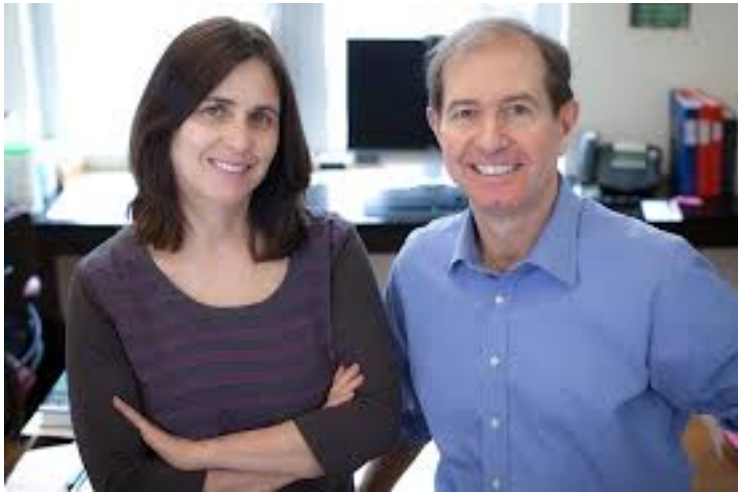
Zero Knowledge Proofs

Let's say Alice proves the Riemann Hypothesis. Can she convince Bob her proof is correct without revealing anything about her proof?

Yes! (if one-way functions exist)

Goldwasser-Micali-Rackof ['85]

Goldreich-Micali-Wigderson ['91]



Convincing Mortals

Let's say God wants to prove his omnipotence by quickly convincing us mortals that white can always win at chess. Can he convince us?

(phrasing from Rudich and Aaronson)

Problem: more possible games than atoms in the universe.

Convincing Mortals

Let's say God wants to prove his omnipotence by quickly convincing us mortals that white can always win at chess. Can he convince us?

(phrasing from Rudich and Aaronson)

by beating Kasparov? (not good enough)

by beating best computers? (still not good enough)

IP=PSPACE

By letting us ask Him some random questions!

Lund, Fortnow, Karloff, Nisan, and Shamir [‘90]

Shamir [’91]

This isn’t as much about chess as it is about zeroes of polynomials over finite fields and error correcting codes.

Just the Tip of the Iceberg

- Theoretical computer science is a mathematical field that has had tremendous impact on the world.
- Full of beautiful, surprising, and philosophically interesting results.
- To learn more in depth, check out our MCS courses or consider TCS graduate school!