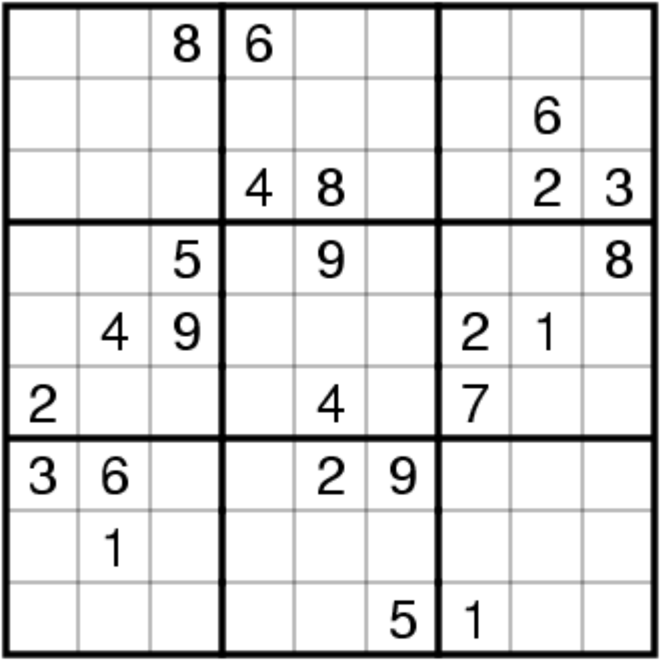
The cosmology of computational problems

Slides by Avi Wigderson

SURVEY

Finding an efficient method to solve SuDoku puzzles is:



		8	6					
							6	
			4	8			2	3
		5		9				8
	4	9				2	1	
2				4		7		
3	6			2	9			
	1							
					5	1		

1: A waste of time

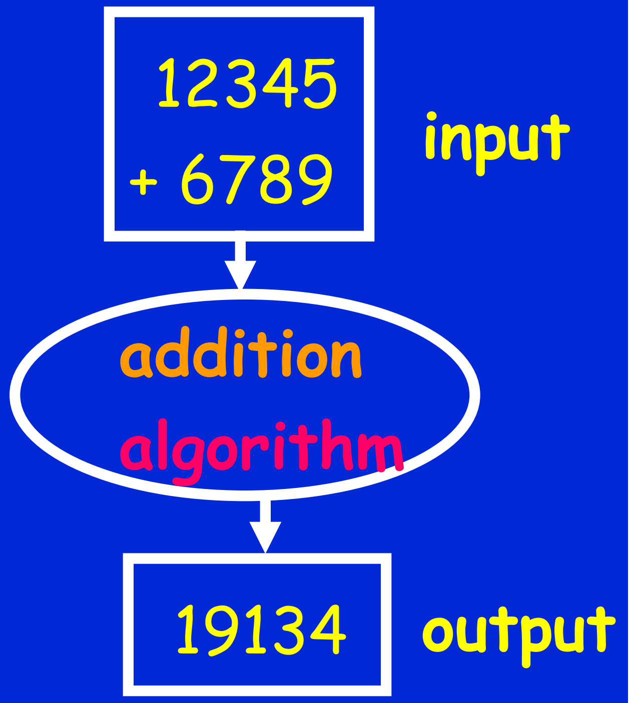
2: A decent way to pass some time

3: A fundamental problem of science and math

Algorithms

Function: input → output

Addition: x,y → x+y



12345

+ 6789

input

addition algorithm

19134

output

ALGORITHM (intuitive def):

Step-by-step, simple procedure, computing a function on *all* inputs

ALGORITHM (Formal def):

Turing machines

Algorithmic solvability

Function: input → output.

Unsolvable: no algorithm halts on all inputs

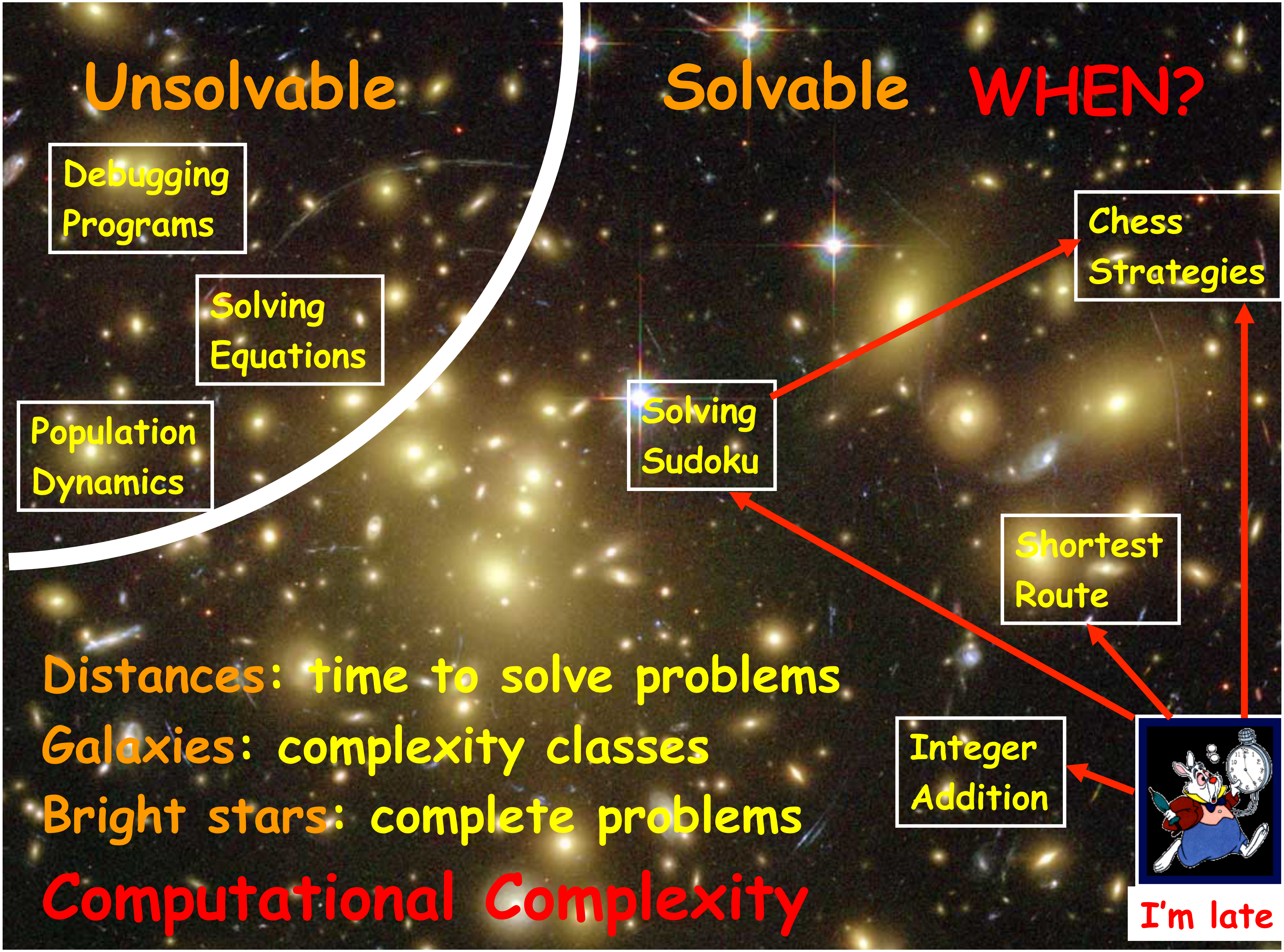
equation → are there integer solutions ?

computer program → is it buggy ?

Solvable: there is a finite algorithm

x,y → x+y

game → does white have a winning strategy ?



Unsolvable

Solvable WHEN?

Debugging Programs

Solving Equations

Population Dynamics

Solving Sudoku

Chess Strategies

Shortest Route

Integer Addition

I'm late

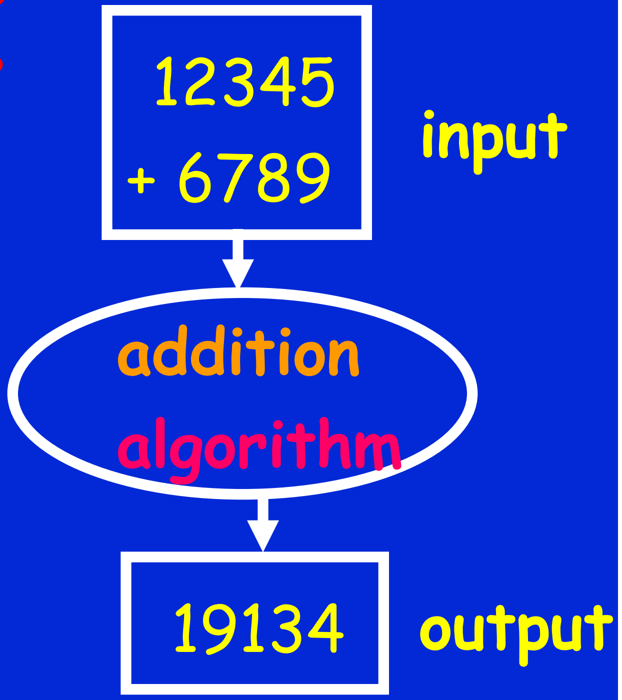
Distances: time to solve problems

Galaxies: complexity classes

Bright stars: complete problems

Computational Complexity

Time complexity I



12345

+ 6789

input

addition algorithm

19134

output

Depends on the implementation?

Technology vs Algorithm

Moore's "law": density and speed doubles every 18 months

Impossibility of exponential growth

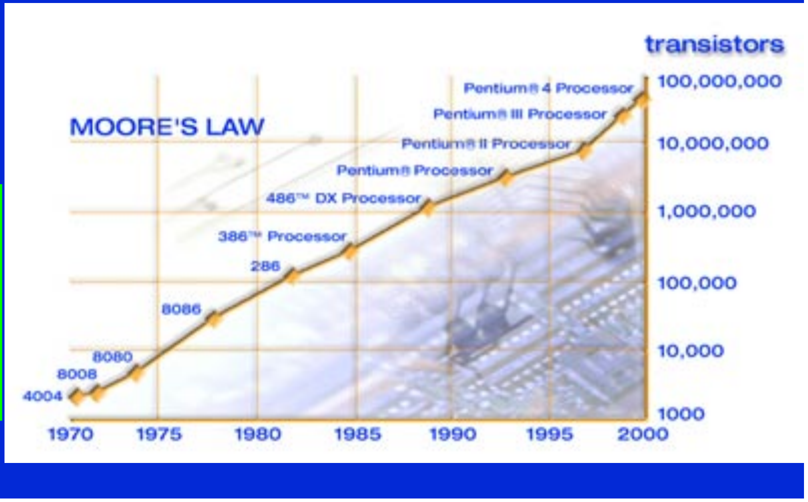
Axiom: transistor ≥ atom

speed ≤ speed of light

Assume we reached this limit!

Time = number of basic steps

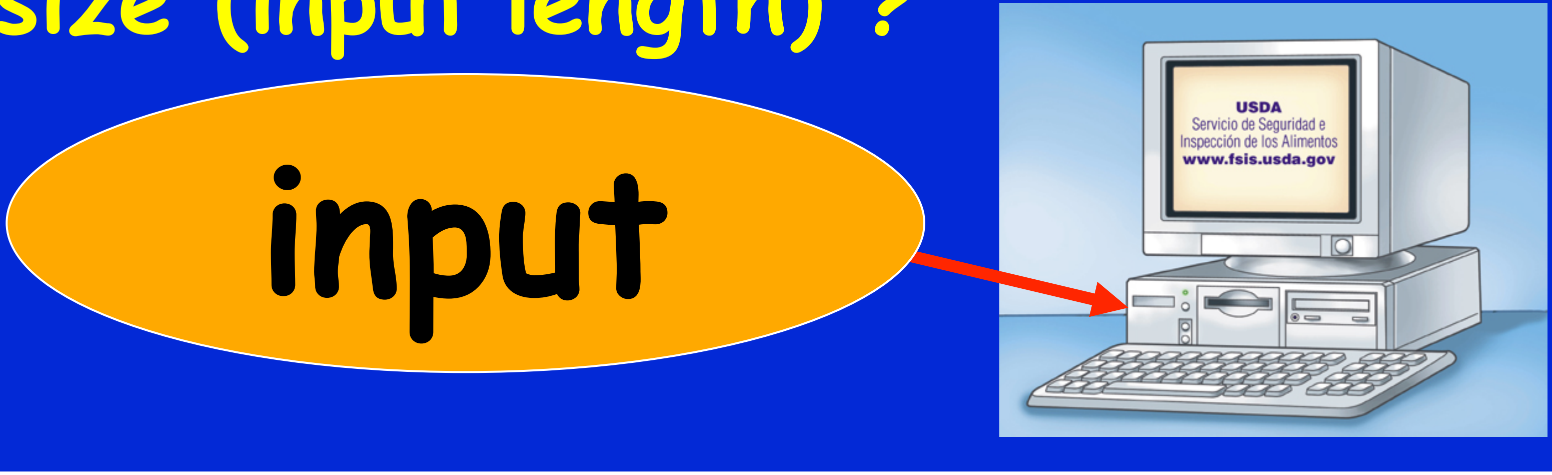
Technology-independent def



Time complexity II

Asymptotic complexity
(of an algorithm)

How does the number of steps of an algorithm increases with the data size (input length) ?



input

Rubik's cube

2

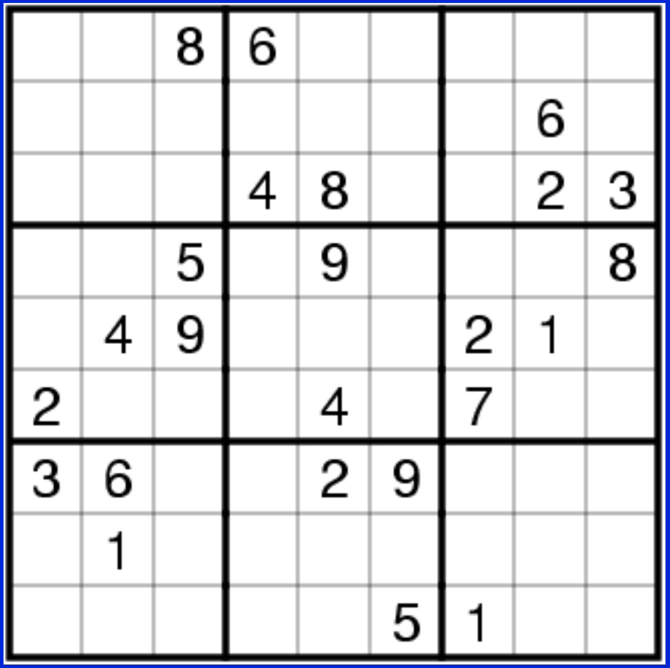
3

4

5

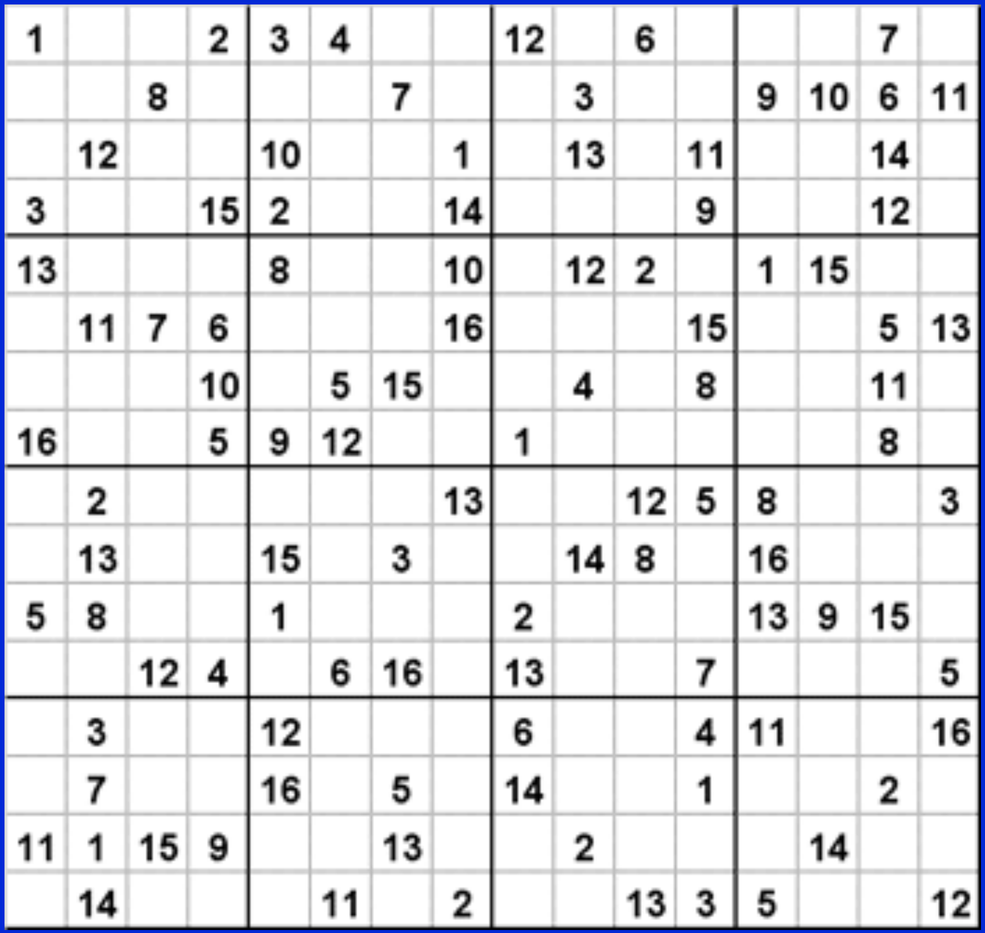
…

Sudoku



		8	6					
							6	
			4	8			2	3
		5		9				8
	4	9				2	1	
2				4		7		
3	6			2	9			
	1							
					5	1		

3



1			2	3	4			12		6				7	
		8				7			3			9	10	6	11
	12			10			1		13		11			14	
3			15	2			14				9			12	
13				8			10		12	2		1	15		
	11	7	6				16				15			5	13
			10		5	15			4		8			11	
16			5	9	12			1						8	
	2						13			12	5	8			3
	13			15		3			14	8		16			
5	8			1				2				13	9	15	
		12	4		6	16		13			7				5
	3			12				6			4	11			16
	7			16		5		14			1			2	
11	1	15	9			13			2				14		
	14				11		2			13	3	5			12

4

Sudoku

	y				b		a	c	x	n			h			t		f	l			d	e	
	t		s	u	j	h	v				d	q	c			o	k		b	n		a	w	p
w	h	e	m	a	n		l	u	k	p		r	y		s	x	d	q	c	o	j		i	b
b		j		p	s			t				i	m		v	n	g	h	a	q		r	x	y
x	o	l	d		i	p			r	e			f		u	j	w	y	m		h		s	c
	w	q	u	j			i		e		x	b	o	m	a			n	h	k	c	s		
n	c				w	x	u	s	f		q		l			e	m	k	v				j	a
a	i		x	f	c	l			m		v	k	w		q			j		d	g		b	h
s			v		h	k	p	o	b	u	f	j	n				t		d	i	m		r	q
	b	d		m	r	v			j		h	p			o	g	y	w			t		u	
y	p		e	l	a	m		v	h	o	b		x	i	t	s	q	u	w	g	r	c	d	k
	q	g	j		e		s	r		h	c				f	k			x		y	l	a	o
		u	t	k		n	o		l		r	m	q	y		b	a	v	j		i	p	h	
	x	r		w	p		y	k	i		l	e	j					m		t	q	v		u
	s		n	b	q	c		g	w	k	a	u	t	p	y		o		r	x		j	m	
j	n	s	q	v	x	y	h		u	t	p	o	g	l	m		f	d			w	i	k	r
u		w	b	t	l	e	r	p	o	m		c	d	f	k	v					s	q		
d			h		m	s	c	f		q	j		k	n	g	w		b		l	v	u		e
			o	e	d	i	k	n	q		w		u		j	a	l			h		b	p	m
l	k				v	j	t	w		a	s	h					u	r	q	c	d	f		n
			g	d	y	r	w			c		l	i		n	p	v	a	f	e			q	
	v	x	p	o		t	b			d	n	f			w			g		s	a	h	y	i
i		k	w	c	g	q	x	h					a	u	l	d	e		s			m	f	v
		a	y	r		d	f	e	n	x	k		s	h			b		u		p			
q	l		f	s			m	i	v			w			h		x	t	y				c	d

5

......

Asymptotic Complexity

# digits	# steps	
	Hindu	
1	6·1	
5	6·5	
10	6·10	
100	6·100	
N	6·N	
N	~N	

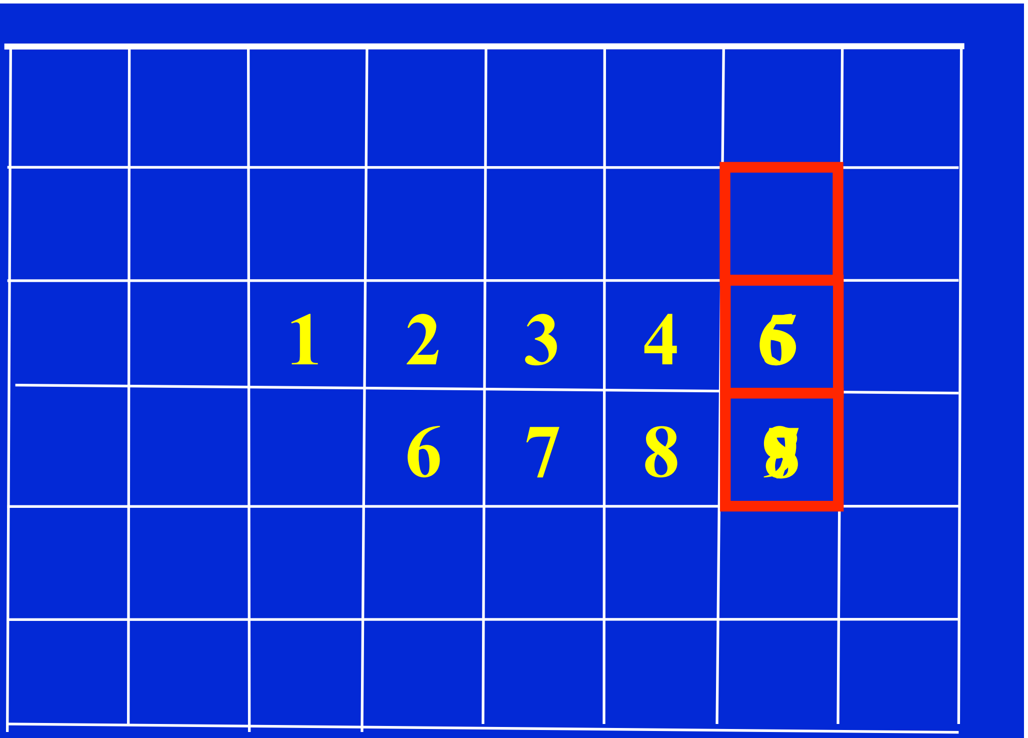
		1	1	1	
	1	2	3	4	5
		6	7	8	9
	1	9	1	3	4

Addition: Hindu algorithm

```
Set i:=0, C:=0
While X[i] and Y[i] nonempty
   W := X[i] + Y[i] + C
   If W>9 then Z[i]:= W-10, C:=1
            else   Z[i]:= W,      C:=0
   i := i+1
endWhile
```

Comparing algorithms

# digits	# steps	
	Hindu	Greek
N	~N	~10^N



Hindu: optimal - “It was the best of

Greek: terrible -“it was the worst of

Complexity of a function =
Complexity of its best algorithm

Addition:
Greek algorithm

```
While Y>0
  Y:=Y-1
  X:=X+1
endWhileh
```

Trivia: power of decimals

10^{80}=10000000000000000000000000000000000000000
00000000000000000000000000000000000000000

≈ number of atoms in the universe

10^{80} - is a small number to write down
- is a large number to count to

10^{40}=10000000000000000000000000000000000000000

≈ number of steps of the fastest
computer before the sun dies

Complexity of functions

comp(add) = n

~~comp(multiply) ≤ n^2 [gradeschool]~~

comp(multiply) ≤ $n\cdot(\log n)$ [schoenhage-strassen]

Is there a better algorithm?

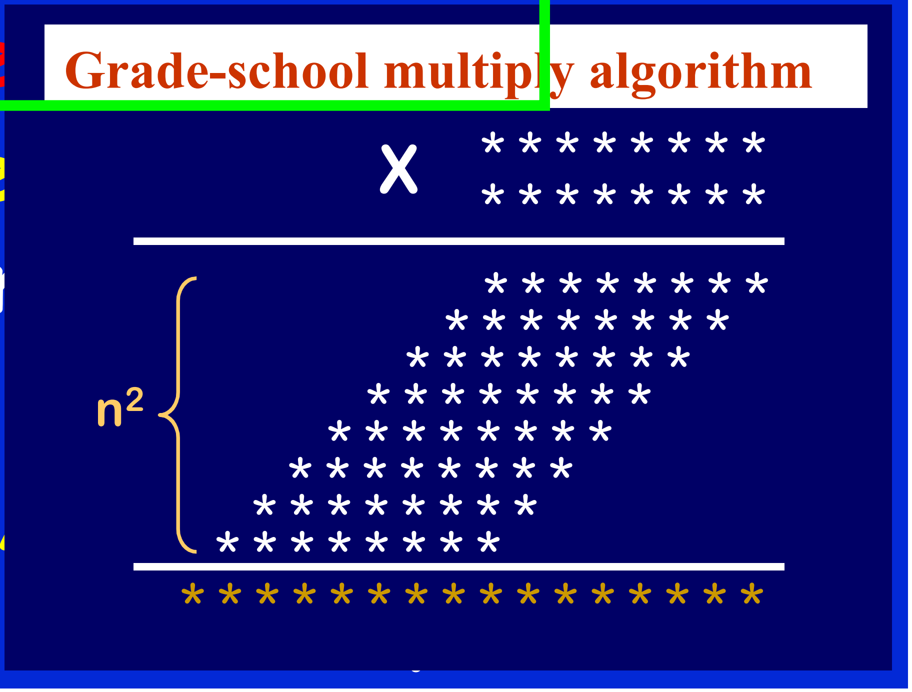
Is there no bette

Main challenges of The

Only efficient algorith

Efficient: n, $n\cdot\log n$, n^2,

Inefficient: 2^n, $2^{\sqrt{n}}$,…



Efficient algorithms - Gems of computer science

Drivers of invention & industry

Who were

Edison ?
Archimedes ?
Guttenberg ?
Bell ?
……

Dijkstra ?
Tukey ?
Berlekamp ?
Knuth ?
……

Few gems: elegance, efficiency, utility

Shortest path

Dijkstra 1959

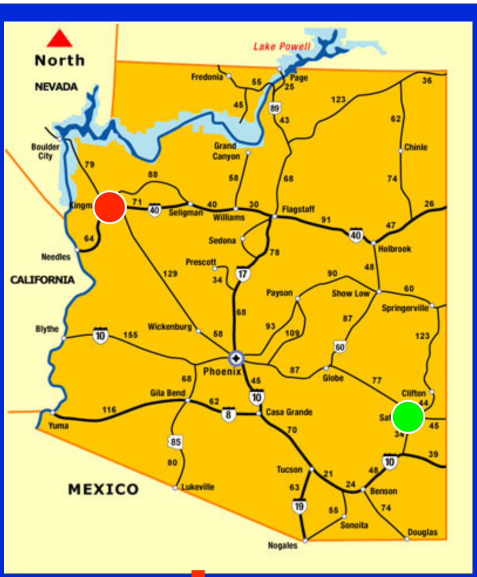
MAPQUEST

Network flows

Internet routing

Dynamic Programming

……



```
define Dijkstra(Graph G, Node s)
          S := {}
          Q := Nodes(G)
          while not empty(Q)
              u := extractMin( Q )
              S := S ∪ u
              for each node v in neighbors( u )
                  if d(u) + w(u,v) < d(v) then
                  d(v) := d(u) + w(u,v)
                  pi(v) := u
```

Distance (Cingman, Safford)

Path (Cingman, Safford)

Pattern matching

Knuth-Morris-Pratt

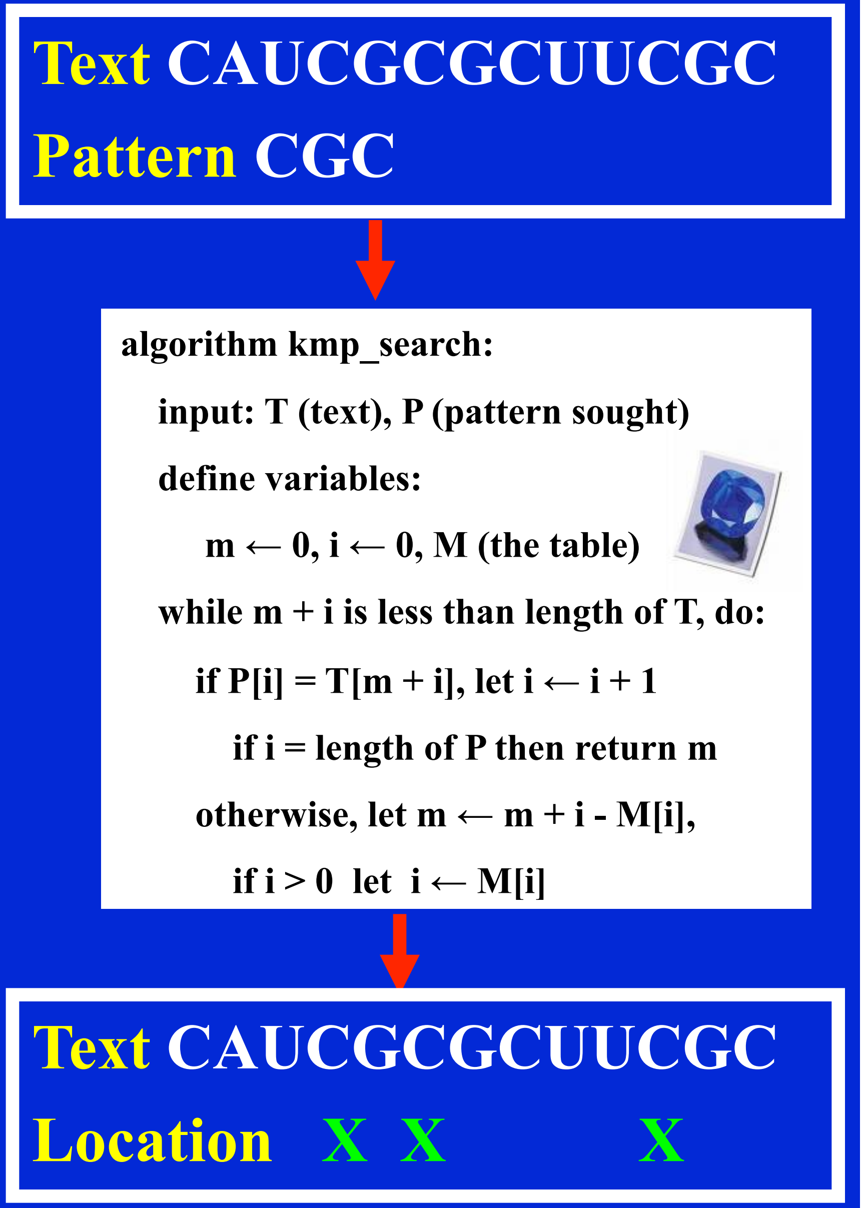
Boyer-Moore 1977

Text processing

Genome

Molecular Biology

Web search



Text CAUCGCGCUUCGC

Pattern CGC

```
algorithm kmp_search:
   input: T (text), P (pattern sought)
   define variables:
       m ← 0, i ← 0, M (the table)
   while m + i is less than length of T, do:
      if P[i] = T[m + i], let i ← i + 1
         if i = length of P then return m
      otherwise, let m ← m + i - M[i],
         if i > 0  let  i ← M[i]
```

Text CAUCGCGCUUCGC

Location X X X

Fast Fourier Transform (FFT)

Cooley-Tukey 1965

Gauss 1805

Audio processing

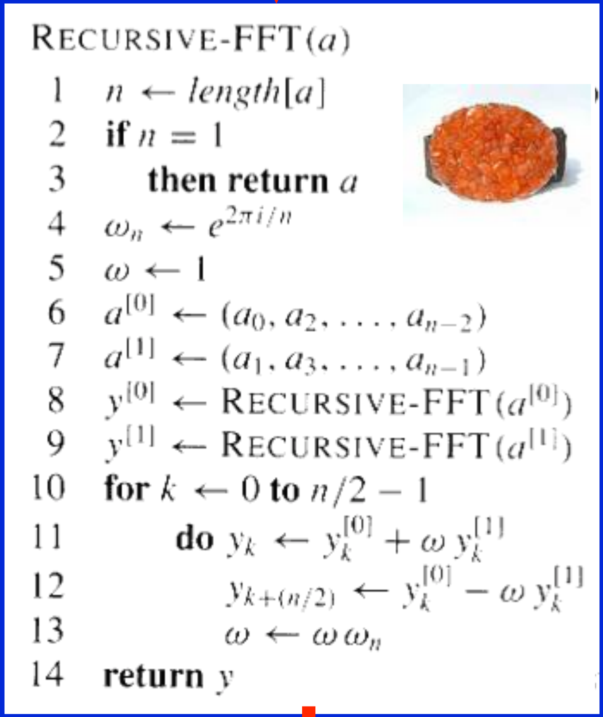
Image processing

Tomography, MRI

Fast multiplication

Quantum algorithms

T(0), T(1), T(2), ….T(N)

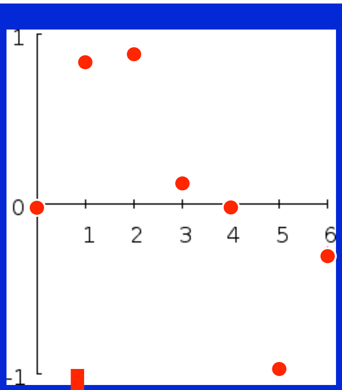


```
RECURSIVE-FFT($a$)
    $n \leftarrow length[a]$
    if $n = 1$
        then return $a$
    $\omega_n \leftarrow e^{2\pi i/n}$
    $\omega \leftarrow 1$
    $a^{[0]} \leftarrow (a_0, a_2, \ldots, a_{n-2})$
    $a^{[1]} \leftarrow (a_1, a_3, \ldots, a_{n-1})$
    $y^{[0]} \leftarrow$ RECURSIVE-FFT($a^{[0]}$)
    $y^{[1]} \leftarrow$ RECURSIVE-FFT($a^{[1]}$)
    for $k \leftarrow 0$ to $n/2 - 1$
        do $y_k \leftarrow y_k^{[0]} + \omega\, y_k^{[1]}$
           $y_{k+(n/2)} \leftarrow y_k^{[0]} - \omega\, y_k^{[1]}$
           $\omega \leftarrow \omega\, \omega_n$
    return $y$
```

$$T_N(x) = \sum_{n=0}^{N} a_n \cos(nx) + \mathrm{i} \sum_{n=0}^{N} a_n \sin(nx)$$

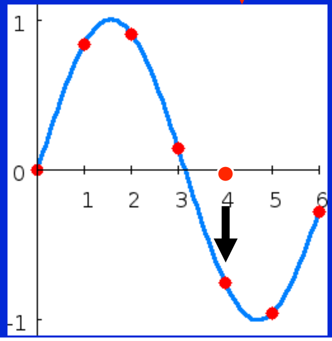
Error correction

Reed-Solomon decoding



Berlekamp-Massey 68

```
INPUT: a binary sequence $S = S_O,S_1,S_2,....S_n$.
OUTPUT: the complexity L (S) of S,   0< L (S) < N.
1. Initialization: C(D):=l, L:=O m:=-l, B{D):=l, N:=O.
2. While (N < n) do the following:
    2.1 Compute the next discrepancy d.
        $d:= (S_N + \Sigma c_i S_{N-i}) \bmod 2$.
    2.2 If d = 1 then do the following:
        $T (D):=C (D), C (D):=C(D)+B(D)\cdot D^{N-m}$.
        If L < N/2 then L:=N+l-L, m:=N, B(B):=T (D).
    2.3 N:=N+l.
3. Return(L) .
```



CDs

DVDs

Satellite communication

Cell phone communication

Cobham, Edmonds
Rabin ~1965

The class P

All problems having an efficient algorithm to *find* solutions
(the galaxy of problems closest to us)

Are all practically interesting problems in P?

Three problems

	Input	Output	Complexity
Factoring integers	1541 $2^{67}-1$	23 ×67 193,707,721 × 761,838,257,287	$\leq 2^{\sqrt{n}}$
Proving theorems	n+"Riemann Hypothesis"	n symbol proof	$\leq 2^n$

Solving Sudoku

		8	6					
							6	
			4	8			2	3
		5		9				8
	4	9				2	1	
2				4		7		
3	6			2	9			
	1							
					5	1		

9	2	8	6	1	3	4	5	7
4	7	3	9	5	2	8	6	1
1	5	6	4	8	7	9	2	3
7	3	5	2	9	1	6	4	8
6	4	9	7	3	8	2	1	5
2	8	1	5	4	6	7	3	9
3	6	7	1	2	9	5	8	4
5	1	2	8	7	4	3	9	6
8	9	4	3	6	5	1	7	2

$\leq n^n$

Verification

$2^{67}-1 = 193707721 \times 761838257287$

n+Poincare Conjectute

n+Fermat's "Theorem"

n = 200 pages

		Input	Output	Complexity
✓	Factoring integers	1541	23 ×67	
		$2^{67}-1$	??	$\leq 2^{\sqrt{n}}$
✓	Proving theorems	n+"Riemann Hypothesis"	n symbol proof	$\leq 2^n$
✓	Solving Sudoku			$\leq n^n$

		8	6					
							6	
			4	8			2	3
		5		9				8
	4	9				2	1	
2				4		7		
3	6			2	9			
	1							
					5	1		

9	2	8	6	1	3	4	5	7
4	7	3	9	5	2	8	6	1
1	5	6	4	8	7	9	2	3
7	3	5	2	9	1	6	4	8
6	4	9	7	3	8	2	1	5
2	8	1	5	4	6	7	3	9
3	6	7	1	2	9	5	8	4
5	1	2	8	7	4	3	9	6
8	9	4	3	6	5	1	7	2

What is common to all 3 problems?

-All look currently intractable, even for moderate n (best algorithms exponential)

- Specific instances get solved!

- Easy verification of given solutions !!!

Cook & Levin
~1971

The class NP

All problems having efficient verification algorithms of given solutions

For every such problem, finding a solution (of length n) takes ≤ 2^n steps: try all possible solutions & verify each.

Can we do better than "brute force" ?
Do all NP problems have efficient algs ?

P versus NP

P: Problems for which solutions can be efficiently *found*

NP: Problems for which solutions can be efficiently *verified*

Fact: $P \subseteq NP$ [finding implies verification]

Conjecture: $P \neq NP$ [finding is much harder than verification]

"P=NP?" is a central question of math, science & technology !!!

what is in NP?

Mathematician: Given a statement, *find* a proof

Scientist: Given data on some phenomena,
find a theory explaining it.

Engineer: Given constraints (size,weight,energy)
find a design (bridge, medicine, phone)

In many intellectual challenges, *verifying* that we found a good solution is an easy task !
(if not, we probably wouldn't start looking)

If P=NP, these have fast, automatic *finder*

How do we tackle P vs. NP?

Break RSA,
ruin E-commerce

Fame & glory
$6M from CLAY

Take out the fun of
Doing these puzzles

	Input	Output	Complexity
Factoring integers	1541 $2^{67}-1$	23 ×67 ??	$\le 2^{\sqrt{n}}$
Proving theorems	n+"Riemann Hypothesis"	n symbol proof	$\le 2^n$
Solving SuDoku	see grid 1 below	see grid 2 below	$\le n^n$

		8	6					
							6	
			4	8			2	3
		5		9				8
	4	9				2	1	
2				4		7		
3	6			2	9			
	1							
					5	1		

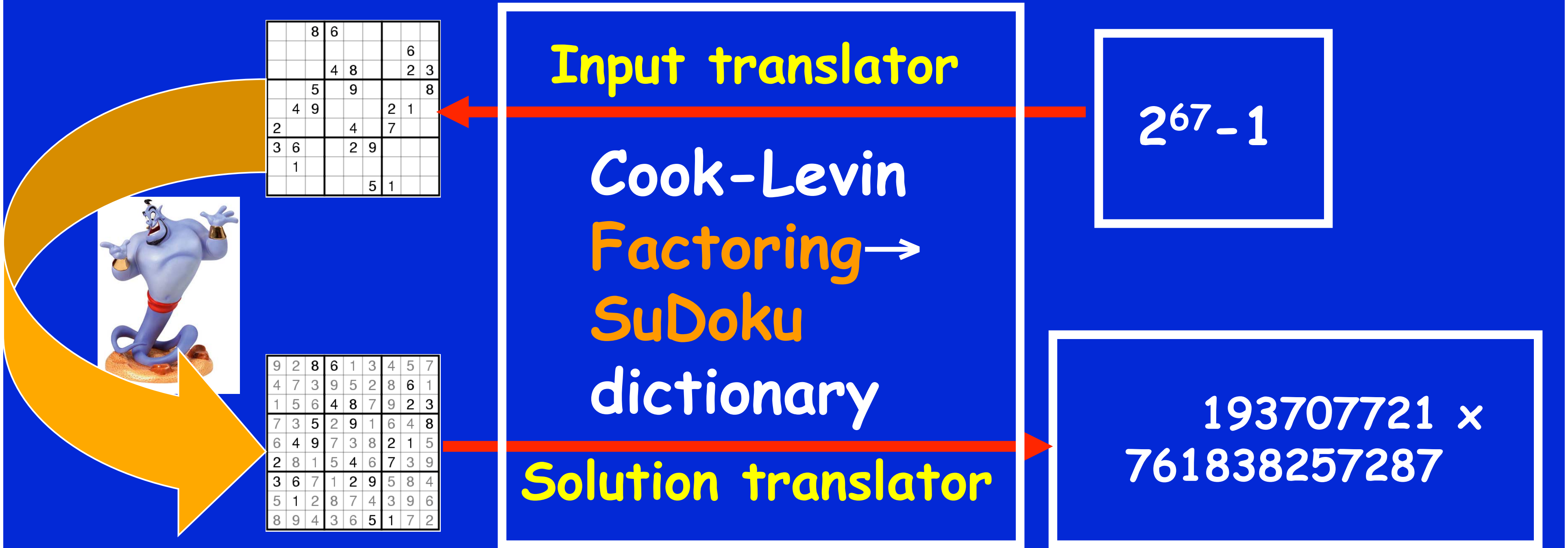
9	2	8	6	1	3	4	5	7
4	7	3	9	5	2	8	6	1
1	5	6	4	8	7	9	2	3
7	3	5	2	9	1	6	4	8
6	4	9	7	3	8	2	1	5
2	8	1	5	4	6	7	3	9
3	6	7	1	2	9	5	8	4
5	1	2	8	7	4	3	9	6
8	9	4	3	6	5	1	7	2

Let's choose the SuDoku solver

Pick any *one* of the three problems.
I'll solve it on each input instantly.
Choose, oh Master!

The power of SuDoku I

Using SuDoku solver for Integer factoring



		8	6					
							6	
			4	8			2	3
		5		9				8
	4	9				2	1	
2				4		7		
3	6			2	9			
	1							
					5	1		

9	2	8	6	1	3	4	5	7
4	7	3	9	5	2	8	6	1
1	5	6	4	8	7	9	2	3
7	3	5	2	9	1	6	4	8
6	4	9	7	3	8	2	1	5
2	8	1	5	4	6	7	3	9
3	6	7	1	2	9	5	8	4
5	1	2	8	7	4	3	9	6
8	9	4	3	6	5	1	7	2

Input translator

Cook-Levin

Factoring→

SuDoku

dictionary

Solution translator

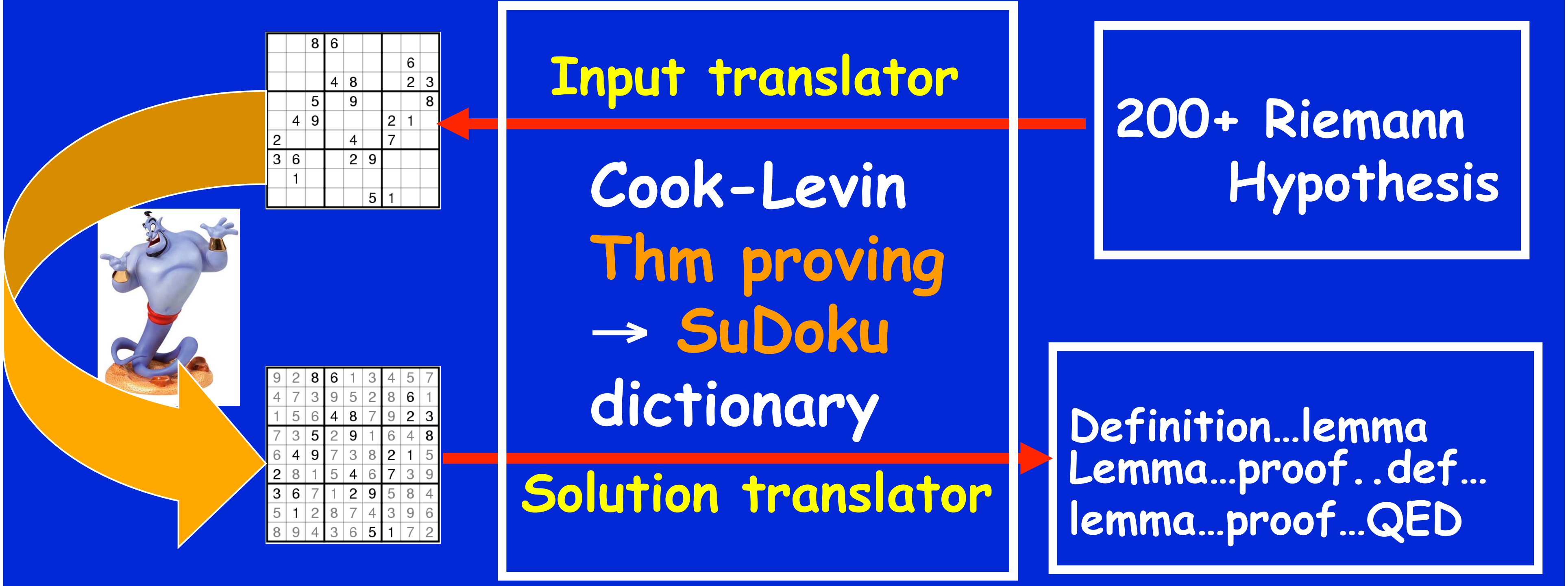
$2^{67}-1$

193707721 ×
761838257287

Both translators are efficient algorithms!

The power of SuDoku II

Using SuDoku solver for Theorem proving



		8	6					
							6	
			4	8			2	3
		5		9				8
	4	9				2	1	
2				4		7		
3	6			2	9			
	1							
					5	1		

Input translator

Cook-Levin
Thm proving
→ SuDoku
dictionary

Solution translator

200+ Riemann
Hypothesis

9	2	8	6	1	3	4	5	7
4	7	3	9	5	2	8	6	1
1	5	6	4	8	7	9	2	3
7	3	5	2	9	1	6	4	8
6	4	9	7	3	8	2	1	5
2	8	1	5	4	6	7	3	9
3	6	7	1	2	9	5	8	4
5	1	2	8	7	4	3	9	6
8	9	4	3	6	5	1	7	2

Definition…lemma
Lemma…proof..def…
lemma…proof…QED

Both translators are efficient algorithms!

Universality: NP-completeness

SuDoku solver can solve any NP problem

Cook-Levin '71: NP-complete problems exist!

SAT is NP-complete. "Meta dictionary" to any NP problem. Another efficient algorithm gem.

Karp '72: NP-complete problems abound!

21 problems in logic, optimization, algebra,..

Today: ~3000 problems in all sciences, *equivalent*

Yato '03 (MSc thesis): SuDoku is NP-complete

P=NP iff SuDoku has an efficient algorithm

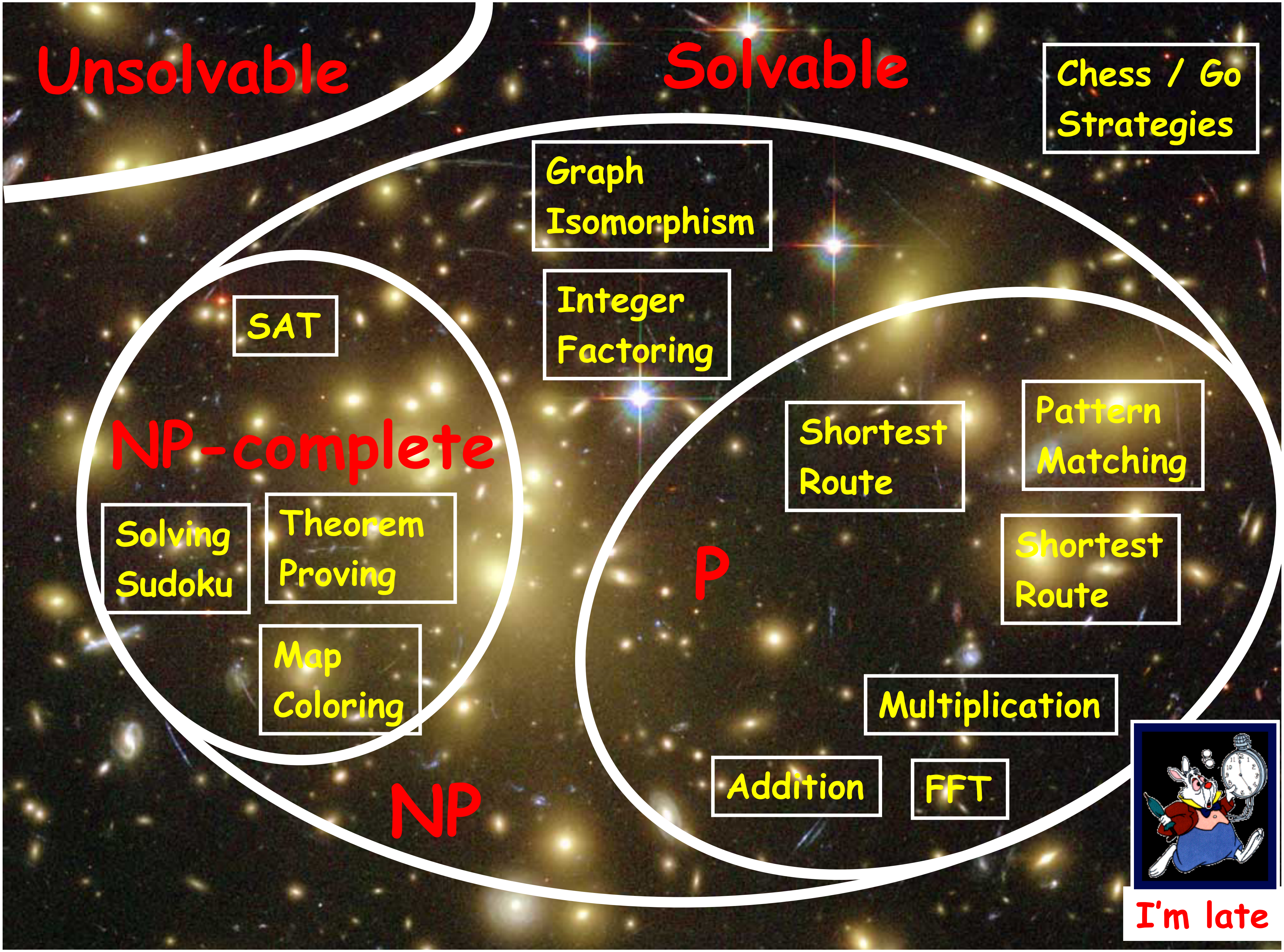
Universality: NP-completeness

NP-complete problems:

If one is easy, then all are!

If one is hard, then all are!

SuDoku,	NP-complete
Thm proving:	NP-complete
Integer factoring:	we don't know



Unsolvable

Solvable

Chess / Go Strategies

Graph Isomorphism

Integer Factoring

SAT

NP-complete

Solving Sudoku

Theorem Proving

Map Coloring

NP

Shortest Route

Pattern Matching

Shortest Route

P

Multiplication

Addition

FFT

I'm late