The P vs. NP problem

Efficient computation, Internet security, and the limits of human knowledge

Avi Wigderson
Institute for Advanced Study
Clay Math Institute Millennium Problems - $1M each

- Birch and Swinnerton-Dyer Conjecture
- Hodge Conjecture
- Navier-Stokes Equations
- P vs. NP
- Poincaré Conjecture
- Riemann Hypothesis
- Yang-Mills Theory
Scientific / Mathematical / Intellectual / Computational problems

NP: Problems we want to solve/understand

P: Problems we can solve/understand

P=NP? - limits on human knowledge
PLAN

- Computation is everywhere
- Algorithms: language of computation
- Efficient algorithms: P
- Efficient verification: NP
- NP-completeness
- Implications
Computation is everywhere

**Computation:** every process which is a sequence of **simple, local** steps, that we want to **perform**, or **understand**

Variety of natural phenomena and intellectual challenges, each with an **essential computational** component.
Fetal development

Weather evolution

Nature computes!
Can we simulate/predict?
SARS infection (in the world) 4/11/03

SARS infection (in the cell) 4/30/03

Will the epidemic spread, or die out?
Solving equations

\[ X^2 + Y^2 = Z^2 \]

X=3 Y=4 Z=5

Proving theorems

\[ X^n + Y^n = Z^n \quad n > 2 \]

Theorem: no solution!
Proof does not fit on this slide (200 pages)

Computations in Mathematics
Haiti earthquake: not enough doctors
BBC News - 9 hours ago
Between 100000 and 200000 people may have died in Haiti as a result of the devastating earthquake that struck last Tuesday.

The subconscious brain computes

Face recognition

"Mona Lisa"

Emotional reactions

Sadness
Beauty from computation

Seashells compute
How to describe computation?

The language of Algorithms
Father of Computing

Alan Turing  1912-1954

1936: “On computable numbers, with an application to the entscheidungsproblem”

- Formal definition of **algorithm** (Turing machine)
- Seed of the computer revolution
- **Church-Turing Thesis**: everything that nature computes, can be emulated on a Turing machine
- Limits on the power of algorithms.
**ALGORITHM (informal)**

Step-by-step, local, simple, mechanical procedure.
Halts in **finite** time for every input.

---

**Example: Addition algorithm (informal)**

1. Scan column. If empty, stop.
2. Add digits. Write answer, remember carry.
3. Move one column left, write carry.
4. Go to 1

---

Finite description vs. Infinite # inputs
Limits of Knowledge I

Unsolvable

Turing (& Godel): Given a computer program, does it always halt?

Mattiasevich: Given an equation, does it have an integer solution?

Conway: Given a (rule for) epidemic, will it spread or die?

Solvable

When?

Computational Complexity Theory
Efficiency of an algorithm - asymptotic analysis:
Number of basic steps, for larger and larger inputs.
Rubik's cube

How many steps to solve..
Sudoku
How long does it take you to solve...

3

4
<table>
<thead>
<tr>
<th></th>
<th>b</th>
<th>a</th>
<th>c</th>
<th>x</th>
<th>n</th>
<th>h</th>
<th>t</th>
<th>f</th>
<th>l</th>
<th>d</th>
<th>e</th>
</tr>
</thead>
<tbody>
<tr>
<td>t</td>
<td>s</td>
<td>u</td>
<td>j</td>
<td>h</td>
<td>v</td>
<td>d</td>
<td>q</td>
<td>c</td>
<td>o</td>
<td>k</td>
<td>b</td>
</tr>
<tr>
<td>w</td>
<td>h</td>
<td>e</td>
<td>m</td>
<td>a</td>
<td>n</td>
<td>l</td>
<td>u</td>
<td>k</td>
<td>p</td>
<td>r</td>
<td>y</td>
</tr>
<tr>
<td>b</td>
<td>j</td>
<td>p</td>
<td>s</td>
<td>t</td>
<td>i</td>
<td>m</td>
<td>v</td>
<td>n</td>
<td>g</td>
<td>h</td>
<td>a</td>
</tr>
<tr>
<td>x</td>
<td>o</td>
<td>l</td>
<td>d</td>
<td>l</td>
<td>p</td>
<td>r</td>
<td>e</td>
<td>f</td>
<td>u</td>
<td>j</td>
<td>w</td>
</tr>
<tr>
<td>w</td>
<td>q</td>
<td>u</td>
<td>j</td>
<td>i</td>
<td>e</td>
<td>x</td>
<td>b</td>
<td>o</td>
<td>m</td>
<td>a</td>
<td>n</td>
</tr>
<tr>
<td>n</td>
<td>c</td>
<td>w</td>
<td>x</td>
<td>u</td>
<td>s</td>
<td>f</td>
<td>q</td>
<td>l</td>
<td>e</td>
<td>m</td>
<td>k</td>
</tr>
<tr>
<td>a</td>
<td>i</td>
<td>x</td>
<td>f</td>
<td>c</td>
<td>l</td>
<td>m</td>
<td>v</td>
<td>k</td>
<td>w</td>
<td>q</td>
<td>j</td>
</tr>
<tr>
<td>s</td>
<td>v</td>
<td>h</td>
<td>k</td>
<td>p</td>
<td>o</td>
<td>b</td>
<td>u</td>
<td>f</td>
<td>j</td>
<td>n</td>
<td>t</td>
</tr>
<tr>
<td>b</td>
<td>d</td>
<td>m</td>
<td>r</td>
<td>v</td>
<td>j</td>
<td>h</td>
<td>p</td>
<td>o</td>
<td>g</td>
<td>y</td>
<td>w</td>
</tr>
<tr>
<td>y</td>
<td>p</td>
<td>e</td>
<td>l</td>
<td>a</td>
<td>m</td>
<td>v</td>
<td>h</td>
<td>o</td>
<td>b</td>
<td>x</td>
<td>i</td>
</tr>
<tr>
<td>q</td>
<td>g</td>
<td>j</td>
<td>e</td>
<td>s</td>
<td>r</td>
<td>h</td>
<td>c</td>
<td>t</td>
<td>k</td>
<td>x</td>
<td>y</td>
</tr>
<tr>
<td>u</td>
<td>t</td>
<td>k</td>
<td>n</td>
<td>o</td>
<td>l</td>
<td>r</td>
<td>m</td>
<td>q</td>
<td>y</td>
<td>b</td>
<td>a</td>
</tr>
<tr>
<td>x</td>
<td>r</td>
<td>w</td>
<td>p</td>
<td>y</td>
<td>k</td>
<td>l</td>
<td>i</td>
<td>e</td>
<td>j</td>
<td>m</td>
<td>t</td>
</tr>
<tr>
<td>s</td>
<td>n</td>
<td>b</td>
<td>q</td>
<td>c</td>
<td>g</td>
<td>w</td>
<td>k</td>
<td>a</td>
<td>u</td>
<td>t</td>
<td>p</td>
</tr>
<tr>
<td>j</td>
<td>n</td>
<td>s</td>
<td>q</td>
<td>v</td>
<td>x</td>
<td>y</td>
<td>h</td>
<td>u</td>
<td>t</td>
<td>p</td>
<td>o</td>
</tr>
<tr>
<td>u</td>
<td>w</td>
<td>b</td>
<td>t</td>
<td>l</td>
<td>e</td>
<td>r</td>
<td>p</td>
<td>o</td>
<td>m</td>
<td>c</td>
<td>d</td>
</tr>
<tr>
<td>d</td>
<td>h</td>
<td>m</td>
<td>s</td>
<td>c</td>
<td>f</td>
<td>q</td>
<td>j</td>
<td>k</td>
<td>n</td>
<td>g</td>
<td>w</td>
</tr>
<tr>
<td>o</td>
<td>e</td>
<td>d</td>
<td>i</td>
<td>k</td>
<td>n</td>
<td>q</td>
<td>w</td>
<td>u</td>
<td>j</td>
<td>a</td>
<td>l</td>
</tr>
<tr>
<td>l</td>
<td>k</td>
<td>v</td>
<td>j</td>
<td>t</td>
<td>w</td>
<td>a</td>
<td>s</td>
<td>h</td>
<td>u</td>
<td>r</td>
<td>q</td>
</tr>
<tr>
<td>g</td>
<td>d</td>
<td>y</td>
<td>r</td>
<td>w</td>
<td>c</td>
<td>l</td>
<td>i</td>
<td>n</td>
<td>p</td>
<td>v</td>
<td>a</td>
</tr>
<tr>
<td>v</td>
<td>x</td>
<td>p</td>
<td>o</td>
<td>t</td>
<td>b</td>
<td>d</td>
<td>n</td>
<td>f</td>
<td>w</td>
<td>g</td>
<td>s</td>
</tr>
<tr>
<td>i</td>
<td>k</td>
<td>w</td>
<td>c</td>
<td>g</td>
<td>q</td>
<td>x</td>
<td>h</td>
<td>a</td>
<td>u</td>
<td>l</td>
<td>d</td>
</tr>
<tr>
<td>a</td>
<td>y</td>
<td>r</td>
<td>d</td>
<td>f</td>
<td>e</td>
<td>n</td>
<td>x</td>
<td>k</td>
<td>s</td>
<td>h</td>
<td>b</td>
</tr>
<tr>
<td>q</td>
<td>l</td>
<td>f</td>
<td>s</td>
<td>m</td>
<td>i</td>
<td>v</td>
<td>w</td>
<td>h</td>
<td>x</td>
<td>t</td>
<td>y</td>
</tr>
</tbody>
</table>

5

...
### Efficiency of the addition algorithm

<table>
<thead>
<tr>
<th>N DIGITS</th>
<th>6N STEPS</th>
<th>Example 1</th>
<th>Example 2</th>
<th>Example 3</th>
<th>Example 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>30 STEPS</td>
<td>12345</td>
<td>+6789</td>
<td>12345</td>
<td>+987654321</td>
</tr>
<tr>
<td>10</td>
<td>60 STEPS</td>
<td>123456789</td>
<td>+987654321</td>
<td>123456789</td>
<td>+987654321</td>
</tr>
<tr>
<td>20</td>
<td>120 STEPS</td>
<td>72635273545786043726</td>
<td>+53827484732625435473</td>
<td>72635273545786043726</td>
<td>+53827484732625435473</td>
</tr>
<tr>
<td>50</td>
<td>300 STEPS</td>
<td>47563739203487456438992305757328576452364568456465744576</td>
<td>+98656092843467546234868431987543210979832865874134653472</td>
<td>47563739203487456438992305757328576452364568456465744576</td>
<td>+98656092843467546234868431987543210979832865874134653472</td>
</tr>
</tbody>
</table>

6 basic steps per column

Is there a faster algorithm? **No!**

Solving is as fast as reading the input
Efficiency of the multiplication algorithm

Grade-school multiply algorithm

```
  x   *   *   *   *   *   *   *   *   *   *   *
```

Fast! Multiply 10,000 digits in a second!

Is there a faster algorithm? Yes!

But not as fast as addition
**Brute force factoring algorithm**

Input: A

• For \(B = 2, 3, \ldots, \sqrt{A}\) do:
  • If \(B\) divides \(A\), return \(B, A/B\)

Very slow! 1000 digits \(\rightarrow\) sun will die before finishing

**Find nontrivial factors of a number \(A\)**

Efficiency of a factoring algorithm

\[
? \times ? = 147,573,952,588,676,412,927
\]

Is there a faster algorithm?

Yes, but still extremely slow!
Which problems are hard to solve?

Addition & Multiplication: Easy

Is Factoring hard?

Finding efficient algorithms, or proving that no such algorithms exist: Bread and butter of our field
The class $P$:

All problems having an efficient (polynomial time, e.g. $n$, $n^2$) algorithm like Addition and Multiplication

Many practical interesting problems in $P$
Efficient algorithms – Drivers of invention & industry

Who were

Edison ?  Marconi ?  Guttenberg ?  Stevenson ?
Light bulb  Radio  Printing press  Steam engine

Dijkstra ?  Tukey ?  Berlekamp ?  Knuth ?
Inventors of important efficient algorithms
Shortest path
Dijkstra 1959

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)
            d(v) := d(u) + w(u,v)
            pi(v) := u

Distance (Delhi, Bangalore)
Path (Delhi, Bangalore)
Pattern matching
Knuth-Morris-Pratt
Boyer-Moore 1977

Spell checking
Text processing
Genome
Molecular Biology
Web search

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
Pattern CGC

Text CAUCGCGCUUCGC
Location X X X X X

Text CAUCGCGCUUCGC
Location X X X X X
Fast Fourier Transform (FFT)

Cooley-Tukey 1965

Gauss 1805

Audio processing
Image processing
Tomography, MRI
Fast multiplication
Quantum algorithms

\[ T(0), T(1), T(2), \ldots, T(N) \]

```
RECURSIVE-FFT(a)
1  n ← length[a]
2  if n = 1
3      then return a
4  \( \omega_n \) ← \( e^{2\pi i/n} \)
5  \( \omega \) ← 1
6  \( a[0] \) ← (\( a_0, a_2, \ldots, a_{n-2} \))
7  \( a[1] \) ← (\( a_1, a_3, \ldots, a_{n-1} \))
8  \( y[0] \) ← RECURSIVE-FFT(a[0])
9  \( y[1] \) ← RECURSIVE-FFT(a[1])
10 for \( k \) ← 0 to \( n/2 - 1 \)
11    do \( y_k \) ← \( y_k^{[0]} + \omega y_k^{[1]} \)
12    \( y_{k+(n/2)} \) ← \( y_k^{[0]} - \omega y_k^{[1]} \)
13    \( \omega \) ← \( \omega \omega_n \)
14 return y
```

\[ T_N(x) = \sum_{n=0}^{N} a_n \cos(nx) + i \sum_{n=0}^{N} a_n \sin(nx) \]
INPUT: a binary sequence $S = S_0, S_1, S_2, \ldots, S_n$.
OUTPUT: the complexity $L(S)$ of $S$, $0 < L(S) < N$.
1. Initialization: $C(D) := 1$, $L := 0$, $m := -1$, $B(D) := L$, $N := O$.
2. While ($N < n$) do the following:
   2.1 Compute the next discrepancy $d$.
      $d := (S_N + \sum c_i S_{N-i}) \mod 2$.
   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($L$).
Computation is everywhere.

Unsolvable

Solvable

Pattern Matching

Shortest Path

Error Correction

FFT

Multiplication

Addition
The class $\mathbf{P}$

All problems having an efficient (polynomial time) algorithm

Many interesting problems in $\mathbf{P}$

Are all interesting problems in $\mathbf{P}$?

What are “interesting” problems?

Cobham, Edmonds
Rabin ~1965
Search problems

Short Path: **FIND** short path from Princeton to LA

Pattern Matching: **FIND** CGC in CAUCGCGCUUCGC

Factoring: **FIND** factors of $147,573,952,588,676,412,927$

Theorem Proving: **FIND** a 200-page proof of the “Riemann Hypothesis”

Sudoku: **FIND** solution of

What is common to all these problems? In all, solutions are easy to **check & verify**!

$$147,573,952,588,676,412,927 = 193,707,721 \times 761,838,257,287$$

**Easy!**

**Hard?**

**Lemma…Proof…Lemma…Proof..**
The class NP-problems like FIND: needle in a haystack

May be hard to find

Always easy to verify
The class \( \text{NP} \)

All problems having efficient verification algorithms of given solutions

Cook & Levin 1971
Gödel 1956
Computation is everywhere. Unsolvable problems such as Integer Factoring and Pattern Matching are distinct from Solvable problems like Shortest Path, Theorem Proving, and Error Correction. Further, NP-hard problems like Solving Sudoku, Theorem Proving, and Multiplication are contained within the NP class, while P problems, such as FFT and Addition, are considered easier to solve.
The class $\text{NP}$

All problems having efficient verification algorithms of given solutions

For every such problem, finding a solution (of length $n$) takes $\leq 2^n$ 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

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)

\[ P = NP \Rightarrow \text{fast, automatic finder: Utopia!} \]

“Creativity” can be efficiently automated!
Universality: NP-completeness

Are SuDoku, Theorem Proving, Factoring hard?
These problems are intimately related!!

Theorem: If SuDoku is easy then
- Theorem proving is easy
- Factoring is easy

Proof: SuDoku is NP-complete
SuDoku solver can solve any NP problem

\[ P = NP \iff \text{SuDoku has an efficient algorithm} \]
Universality: \( \text{NP-complete} \) completeness

\( \text{NP-complete problems:} \)
If one is easy, then all are!
If one is hard, then all are!

SuDoku: \( \text{NP-complete} \)
Thm proving: \( \text{NP-complete} \)
Integer factoring: we don't know
Computation is everywhere

Unsolvable

NP-complete

Integer Factoring

Solving Sudoku

Theorem Proving

Solving Sudoku

Theorem Proving

NP

P

Shortest Path

Pattern Matching

 FFT

Error Correction

Multiplication

Addition

Solvable

NP

- complete

Addition

Multiplication

FFT

Error Correction

Pattern Matching

Shortest Path
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

Thousands of NP-complete problems known in Math, Biology, Physics, Economics, ....
The protein threading problem with sequence amino acid interaction preferences is **NP-complete**

Richard H. Lathrop

Finding a Nash equilibrium in spatial games is **NP-complete**

R. Baron, J. Durieu, H. Haller and P. Solal

Quadratic equations over free groups are **NP-complete**

O. Kharlampovich, I.G. Lysenok, A.G. Myasnikov, N. Touikan

**NP-completeness**: sign of structural “nastiness”. Potential guide to better models and theories
P ≠ NP as a law of nature

NP-complete problems that “nature solves”

**Biology:** Minimum energy

Protein Folding

**Physics:** Minimum surface area Foam

**Economics:** Nash Equilibrium in strategic games

Possibilities:
model is wrong or inputs are special or P=NP
What is efficient computation?

**Church-Turing Thesis:**
Every *reasonable* process, can be efficiently simulated by a Turing machine.
- Adding random bits

**Theorem [Blum-Micali, Yao, Nisan-Wigderson, Impagliazzo-Wigderson]**
If “P≠NP”, randomness add no power!
- Adding quantum bits

**Theorem [Shor]**
An efficient algorithm for Factoring
Some of the problems we want to solve are hard. Are hard problems useful?

Cryptography: If Factoring is hard then:
- Encryption
- Digital signatures
- Secure e-mail
- Electronic commerce
- On-line shopping
- Poker by telephone
Things we didn't cover

- How to prove NP-completeness
- Attempts to prove $P \neq NP$ and restricted lower bounds
- Other resources (space, parallelism communication) and complexity classes
- Other modes of computation (average-case, approximate,...)
- ......
Computation is everywhere

Unsolvable

NP-complete

SAT

Solving Sudoku

Theorem Proving

Map Coloring

Solving Sudoku

NP

NP-complete

Integer Factoring

Shortest Path

FFT

Error Correction

Multiplication

Addition

Solvable

Chess / Go Strategies

Theorem Proving

Map Coloring

P

QP

Pattern Matching

Multiplication

Addition

FFT

Error Correction

Shortest Path

Integer Factoring

NP

P