The P vs. NP question, NP-Completeness

Steve Tanimoto
Winter 2016

This lecture material represents the work of multiple instructors at the University of Washington. Thank you to all who have contributed!

The $1M question

The Clay Mathematics Institute
Millenium Prize Problems

1. Birch and Swinnerton-Dyer Conjecture
2. Hodge Conjecture
3. Navier-Stokes Equations
4. P vs NP
5. Poincaré Conjecture
6. Riemann Hypothesis
7. Yang-Mills Theory

The P versus NP problem

Is one of the biggest open problems in computer science (and mathematics) today

It’s currently unknown whether there exist polynomial time algorithms for NP-complete problems
– That is, does P = NP?
– People generally believe P ≠ NP, but no proof yet

But what is the P-NP problem?

Sudoku

3x3x3

Sudoku

3x3x3

Sudoku

4x4x4

Sudoku

4x4x4
Suppose you have an algorithm $S(n)$ to solve $n \times n \times n$

$V(n)$ time to verify the solution

Fact: $V(n) = O(n^2 \times n^2)$

Question: is there some constant such that $S(n) = O(n^{\text{constant}})$?

The $P$ versus $NP$ problem (informally)

Is finding an answer to a problem much more difficult than verifying an answer to a problem?

The Set “HAM”

$HAM = \{ \text{graph } G \mid G \text{ has a Hamiltonian cycle} \}$

The Set “SAT”

$SAT = \{ \text{all satisfiable circuits } C \}$
Sudoku

Input: \( n \times n \times n \) sudoku instance
Output: YES if this sudoku has a solution
NO if it does not

The Set “SUDOKU”
SUDOKU = { All solvable Sudoku instances }

Polynomial Time and The Class “P”

What is an efficient algorithm?

Is an O(n) algorithm efficient?
How about O(n \log n)?
O\left(n^2\right)?
O\left(n^{10}\right)?
O(2^n)?
O(n!)?

What is an efficient algorithm?

Does an algorithm running in \( O(n^{100}) \) time count as efficient?

Asking for a poly-time algorithm for a problem sets a (very) low bar when asking for efficient algorithms.

We consider non-polynomial time algorithms to be inefficient.

And hence a necessary condition for an algorithm to be efficient is that it should run in poly-time.

The Class P

The class of all sets that can be verified in polynomial time.
AND

The class of all decision problems that can be decided in polynomial time.

P

The question is: can we achieve even this for

HAM?
SAT?
Sudoku?
Onto the new class, NP
(Nondeterministic Polynomial Time)

Verifying Membership
Is there a short "proof" I can give you to verify that:
- $G \in \text{HAM}$?
- $G \in \text{Sudoku}$?
- $G \in \text{SAT}$?

Yes: I can just give you the cycle, solution, circuit.

The Class NP
The class of sets for which there exist "short" proofs of membership (of polynomial length) that can "quickly" verified (in polynomial time).

Recall: The algorithm doesn't have to find the proof; it just needs to be able to verify that it is a "correct" proof.

Fact: $P \subseteq NP$

Summary: $P$ versus $NP$
$NP$: "proof of membership" in a set can be verified in polynomial time.
$P$: in $NP$ (membership verified in polynomial time)
AND membership in a set can be decided in polynomial time.

Fact: $P \subseteq NP$

Question: Does $NP \subseteq P$?
i.e., Does $P = NP$?
People generally believe $P \neq NP$, but no proof yet.

Why Care?
**NP Contains Lots of Problems We Don’t Know to be in P**

Classroom Scheduling
Packing objects into bins
Scheduling jobs on machines
Finding cheap tours visiting a subset of cities
Finding good packet routings in networks
**Decryption**
...

OK, OK, I care...

---

**How could we prove that NP = P?**

We would have to show that every set in NP has a polynomial time algorithm...

How do I do that?
It may take a long time!
Also, what if I forgot one of the sets in NP?

---

**Theorem (Cook/Levin)**

SAT is one problem in NP, such that if we can show SAT is in P, then NP \(\subseteq P\).

SAT is a problem in NP that can capture all other languages in NP.

We say SAT is NP-complete.

---

**Poly-time reducible to each other**

Any problem in NP can be reduced (in polynomial time) to an instance of SAT.

SAT can be reduced (in polytime) to an instance of Sudoku.

hence SAT is NP-complete

hence Sudoku is NP-complete

---

**NP-complete: The “Hardest” problems in NP**

- **Sudoku**
- **Clique**
- **Independent-Set**
- **3-Colorability**
- **HAM**

These problems are all “polynomial-time equivalent” i.e., each of these can be reduced to any of the others in polynomial time

If you get a polynomial-time algorithm for one, you get a polynomial-time algorithm for ALL.
(you get millions of dollars, you solve decryption, ... etc.)