Mathematical Puzzles, Games and Other Diversions

Day 8

Derrick Chung

Feb 11, 2021

P vs NP - History

- The underlying issues were first brought up in the 1950's by John Nash and Kurt Gödel.
- First formalized in 1971 by Stephen Cook (and independently by Leonid Levin in 1973).
- Arguably the most important open problem in all of computer science.
- One of Seven Millenium Prize Problems announced by the Clay Mathematics Institute in 2000 ($1 million award).

P vs NP - Formalities

Definition

A **decision problem** is a problem that can be posed as a yes/no question.

Definition

The complexity class **P** is the set of all decision problems that can be solved in polynomial time on a Turing machine.

Definition

The complexity class **NP** is the set of all decision problems whose solutions can be **verified** in polynomial time on a Turing machine.

P vs NP - In Plain English

- A Turing machine is a theoretical model of a computer.
- We use complexity classes to describe how "easy" or "hard" a problem is to solve.
- In this case, we're interested in time. So for the most part, polynomial time just means *quickly* (sort of).
- More specifically, complexity is about scale (or growth) with respect to the size of its input.

Easy Problems (in P)

Addition, Multiplication, primality, sorting

Hard Problems

Perfect play in (generalized) Checkers, Chess or Go

P vs NP - What about NP?

Reminder:

NP is the class of decision problems where **yes** answers can be checked quickly (in polynomial time).

- Every problem in P is in NP, i.e. $P \subseteq NP$.
- Perfect play in Checkers, Chess and Go are not (EXPTIME).

Question:

Does $P = NP$?

If problems can be verified quickly, does it also mean that they can be solved quickly?

P vs NP - Problems in NP

Examples

- Graph Isomorphism
- Factoring
- Discrete Logarithm
- Traveling Salesman Problem, Integer Programming, Minesweeper, Sudoku, Rubik's Cube, Boolean Satisfiability, Graph Colouring, Vertex Cover, Protein Structure, and more. **(NP-Complete)**

Definition

An NP problem is considered NP-complete, if any other NP problem can be *efficiently* transformed into a case of this one.

P vs NP - What's the point?

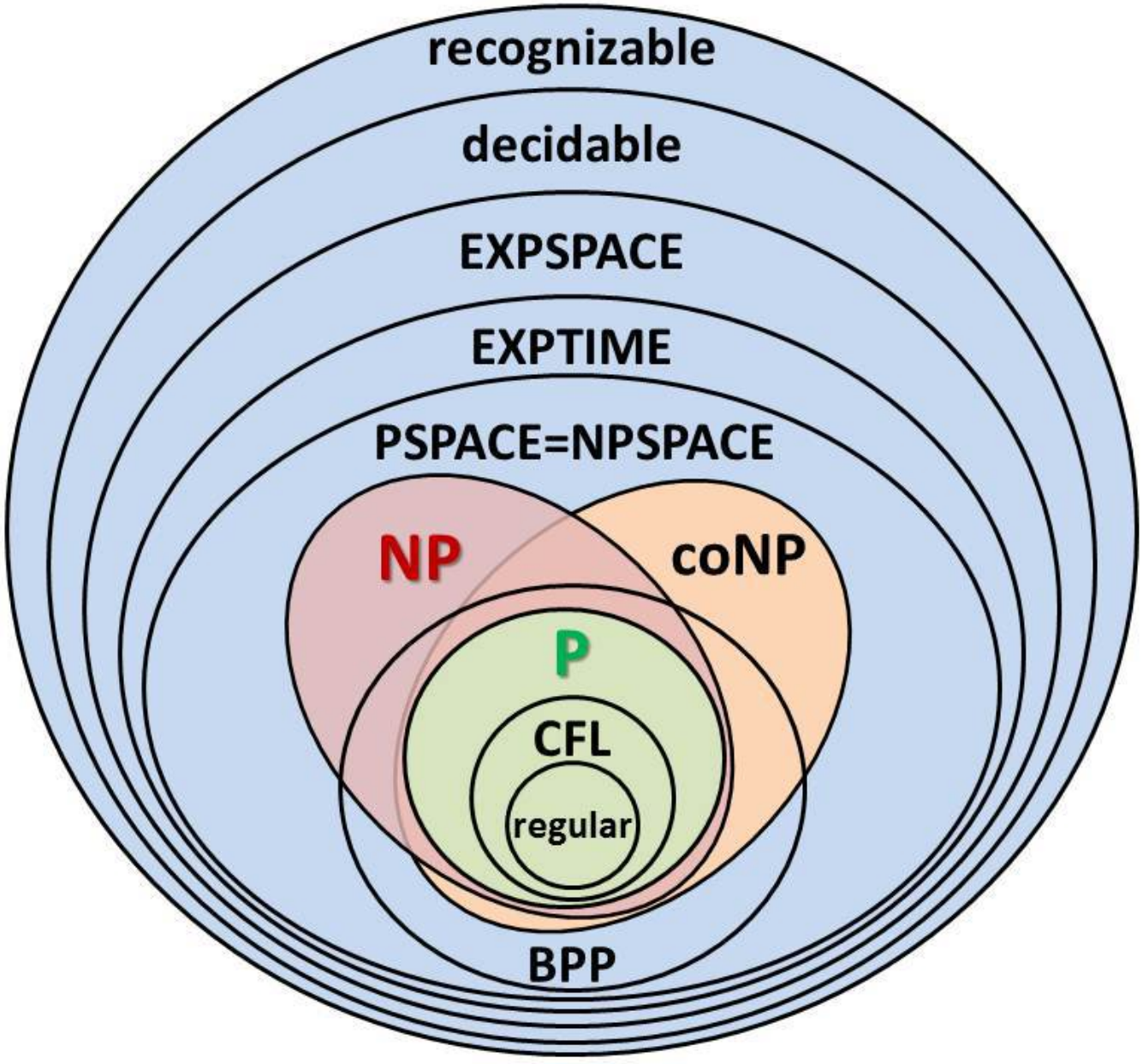
IF P=NP

- *Efficient* algorithms to solve seemingly difficult problems.
- Examples: Scheduling, optimizations, language translation, weather predictions, proteins, factoring
- Easier to design drugs, treat diseases
- Faster Computers
- We'd likely have to revise cryptographic protocols.

IF P $\neq$ NP

- Almost all computer scientists believe that this to be true.
- We can feel safer about our internet security.
- Whatever technique used would be revolutionary.

P vs NP - What's the point? (cont.)



P vs NP - Parting Thoughts

If $P = NP$, then the world would be a profoundly different place than we usually assume it to be. There would be no special value in "creative leaps," no fundamental gap between solving a problem and recognizing the solution once it's found. Everyone who could appreciate a symphony would be Mozart; everyone who could follow a step-by-step argument would be Gauss; everyone who could recognize a good investment strategy would be Warren Buffett.

-Scott Aaronson

The main argument in favor of $P \neq NP$ is the total lack of fundamental progress in the area of exhaustive search. This is, in my opinion, a very weak argument. The space of algorithms is very large and we are only at the beginning of its exploration,... The resolution of Fermat's Last Theorem also shows that very simple questions may be settled only by very deep theories.

-Moshe Vardi