Great Ideas in Theoretical CS

Lecture 17:
P vs. NP

Anil Ada
Ariel Procaccia (this time)

Millennium Prize Problems

- Seven famous problems in math stated in 2000 by the Clay Foundation
- $1,000,000 prize for solving any of them
- One of the problems: **P** vs. **NP**

Millennium Prize Problems

Keith Devlin

If one is solved in the next few years, it'll probably be **P** vs. **NP**

If, in the year 3000, one of them is unsolved, it will be **P** vs. **NP**

Laszlo Lovasz

Millennium Prize Problems

- The **P** vs. **NP** problem is the only Millennium Prize problem that has the potential to change the world
- So what is it?

Sudoku

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

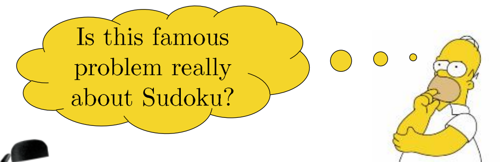
$3 \times 3 \times 3 \times 3$

Sudoku

- Sudoku: Given a partially filled $n \times n \times n \times n$ Sudoku board, can it be filled?
- Naive decision algorithm: Check all possibilities, in time $O\left(n^{2n^4}\right)$
- Verifying a solution: $O(n^4)$
- For $n = 100$
  - Verifying a solution: 100M steps
  - Deciding YES/NO: Number with 400M digits!

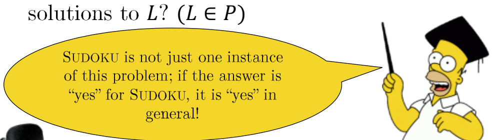
Sudoku

- Question: Is there a polynomial-time algorithm that can solve Sudoku?
- This is equivalent to the **P** vs. **NP** problem!



P vs. NP

- Informal formulation of **P** vs. **NP**:
  - Let L be an algorithmic task
  - Suppose there is an efficient algorithm for verifying solutions to L $(L \in NP)$
  - Is there an efficient algorithm for finding solutions to L? $(L \in P)$

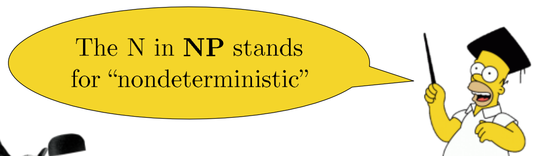


Efficiency

- Efficient = polynomial time
- Given a decision problem L, $x \in L$ means that x is a YES instance of L; $|x|$ is its size
- **P** = Decision problems L such that there exists a constant c and an algorithm A such that A runs in time $|x|^c$ and $A(x) =$ YES if and only if $x \in L$
- We saw last time that 2-Coloring is in **P**

Verifying solutions

- In problems like SUDOKU, verifying the solution can be done efficiently
- **NP** = Decision problems whose solutions can be verified in polynomial time in their input size

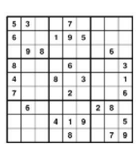


NP: semi-formal definition

- $L \in$ **NP** if and only if there are constants c, d and an algorithm V called the verifier such that:
  - V takes two inputs, x and y, where $|y| \leq |x|^d$; x is called the instance and y is called the certificate
  - $V(x, y)$ runs in time $O(|x|^c)$
  - If $x \in L$, $\exists y$ such that $V(x, y) =$ YES
  - If $x \notin L$, $\forall y$, $V(x, y) =$ NO

Examples

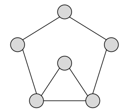
- SUDOKU: Given a partially filled $n \times n \times n \times n$ Sudoku board, can it be completed?
- Input size: n
- Certificate: board filled with numbers
- Verifier: Check that each square, row, and column contain all numbers


Instance

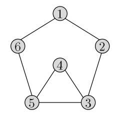

Certificate

Examples

- Hamiltonian-cycle: Given a graph $G = (V, E)$, does it contain a Hamiltonian cycle?
- Input size: $n = |V|$
- Certificate: A permutation of the n vertices
- Verifier: Check that the permutation contains each vertex exactly once, and there is an edge between adjacent vertices



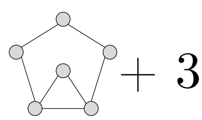
Instance



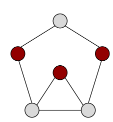
Certificate

Examples

- Independent-set: Given a graph $G = (V, E)$ and $k \in \mathbb{N}$, does G contain an independent set of size k?
- Input size: $n = |V|$
- Certificate: k vertices
- Verifier: Check that there are no edges between pairs of vertices



Instance



Certificate

Examples

- Poll 1: Which of the following two problems is in **NP**?
  1. Given numbers $a_1, \dots, a_n$ and $k \in \mathbb{N}$, is there a subset S such that $\sum_{i \in S} a_i = k$?
  2. Given a graph G and $k \in \mathbb{N}$, is the largest clique of size at most k?
  3. Both
  4. Neither

Examples

- Poll 2: Which of the following two problems is in **NP**?
  1. Given a graph G, does it not have a 2-coloring?
  2. Given a graph G, does it not have an Eulerian cycle?
  3. Both
  4. Neither

P vs. NP

- Theorem: $\mathbf{P} \subseteq \mathbf{NP}$
- Proof:
  - Suppose $L \in \mathbf{P}$
  - Let A be a poly-time algorithm that decides L
  - The verifier V takes as input the instance x and an empty certificate y
  - $V(x, y)$ outputs $A(x)$ ■

P vs. NP

- We know that $\mathbf{P} \subseteq \mathbf{NP}$; does $\mathbf{P} = \mathbf{NP}$?
- If $\mathbf{P} = \mathbf{NP}$ then there would be an efficient algorithm for SUDOKU, 3-COLORING, CIRCUIT-SAT... Awesome!
- If $\mathbf{P} \neq \mathbf{NP}$ then there is some particular $L \in \mathbf{NP}$ such that $L \notin \mathbf{P}$; but maybe it is an obscure L?

THE COOK-LEVIN THEOREM

- Theorem (Cook 71, Levin 73): **P** = **NP** if and only if CIRCUIT-SAT ∈ **P**
- In particular, if **P** ≠ **NP** then CIRCUIT-SAT ∉ **P**
- In a sense, CIRCUIT-SAT is the hardest problem in **NP**

REDUCTIONS, REVISITED

- L has a polynomial-time reduction to L', denoted $L \leq_T^P L'$, if and only if it is possible to solve L in polynomial time using a polynomial-time algorithm for L'
- If $L \leq_T^P L'$ then:
  1. $L' \in \mathbf{P} \Rightarrow L \in \mathbf{P}$
  2. $L \notin \mathbf{P} \Rightarrow L' \notin \mathbf{P}$

THE HARDEST PROBLEM(S)

- If CIRCUIT-SAT is in **P** then all of **NP** is in **P**
- Last lecture: there is a poly-time reduction from CIRCUIT-SAT to 3-COLORING

  ⇒ If 3-COLORING is in **P** then CIRCUIT-SAT is in **P**, and hence all of **NP** is in **P**

  ⇒ **P** = **NP** if and only if 3-COLORING ∈ **P**

The hardest problem(s)

- Theorem (Yato-Seta 2002): There is a poly-time reduction from 3-COLORING to SUDOKU

  ⇒ If SUDOKU is in **P** then 3-COLORING is in **P**, and hence all of **NP** is in **P**

  ⇒ **P** = **NP** if and only if SUDOKU ∈ **P**

Cook-Levin, revisited

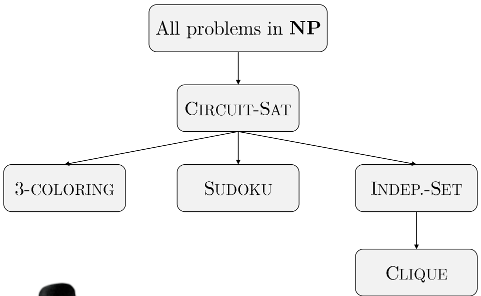
- Actual statement of Cook-Levin: Let L ∈ **NP**, then there is a poly-time reduction from L to CIRCUIT-SAT

  CIRCUIT-SAT ∈ **P** ⇒ **P** = **NP**
  P = **NP** ⇒ CIRCUIT-SAT ∈ **P**

NP-completeness

- L is **NP**-hard if every problem in **NP** has a polynomial time reduction to L
- L is **NP**-complete if L ∈ **NP** and L is **NP**-hard
- To show that a problem is **NP**-complete:
  - Show that it is in **NP**
  - Show that a known **NP**-hard problem reduces to it

NP-COMPLETENESS

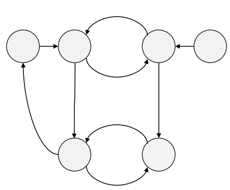


NP-COMPLETE PROBLEMS

- Tens of thousands of problems are known to be **NP**-complete
- If even one of them has a poly-time algorithm then all of them are in **P**

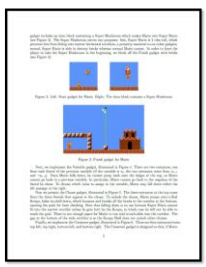
NP-COMPLETE PROBLEMS

- CYCLE-COVER: Given a directed graph and $L \in \mathbb{N}$, is there a collection of disjoint cycles of length $\leq L$ that covers $\geq k$ vertices?
- Theorem: CYCLE-COVER is **NP**-complete
- Relevant to kidney exchange



NP-complete problems

NP-complete problems

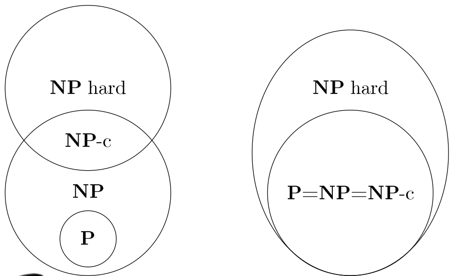


P vs. NP

- So what do the experts think about the P vs. NP problem?
- Two polls from 2002 and 2012
  - 100 respondents in 2002
  - 152 respondents in 2012

Year	P≠NP	P=NP	Ind.	DC	BM
2002	61%	9%	4%	1%	22%
2012	83%	9%	3%	3%	1%

The two possible worlds



Complexity universe

Summary

- Terminology / facts
  - **P** and **NP**
  - Cook-Levin Theorem
  - **NP**-complete
- Principles:
  - Proving that problems are in **P**, **NP**, or **NP**-complete