Lecture 17

Thursday, July 13, 2017 1:10 PM

Decision problems

Yes/No answers

Decision problem.

- Problem X is a set of strings.
- Instance s is one string.
- Algorithm A solves problem X: $A(s) = yes$ iff $s \in X$.

Def. Algorithm A runs in polynomial time if for every string s, $A(s)$ terminates in at most $p(|s|)$ "steps", where $p(\cdot)$ is some polynomial function.

length of s

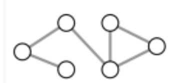
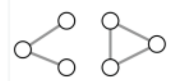
Ex.

- Problem PRIMES = { 2, 3, 5, 7, 11, 13, 17, 23, 29, 31, 37, …. }.
- Instance $s = 592335744548702854681$.
- AKS algorithm: solves PRIMES in $O(|s|^8)$ steps.

4

Definition of P

P. Decision problems for which there is a poly-time algorithm.

Problem	Description	Algorithm	yes	no
MULTIPLE	Is x a multiple of y ?	grade-school division	51, 17	51, 16
REL-PRIME	Are x and y relatively prime ?	Euclid (300 BCE)	34, 39	34, 51
PRIMES	Is x prime ?	AKS (2002)	53	51
EDIT-DISTANCE	Is the edit distance between x and y less than 5 ?	dynamic programming	niether neither	acgggt ttttta
L-SOLVE	Is there a vector x that satisfies $Ax = b$?	Gauss-Edmonds elimination	$\begin{bmatrix} 0 & 1 & 1 \\ 2 & 4 & -2 \\ 0 & 3 & 15 \end{bmatrix}, \begin{bmatrix} 4 \\ 2 \\ 36 \end{bmatrix}$	$\begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 1 \end{bmatrix}, \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$
U-CONN	Is an undirected graph G connected?	depth-first search (Theseus)		

5

NP

Certification algorithm intuition.

- Certifier views things from "managerial" viewpoint.
- Certifier doesn't determine whether $s \in X$ on its own; rather, it checks a proposed proof t that $s \in X$.

Def. Algorithm $C(s, t)$ is a certifier for problem X if for every string s, $s \in X$ iff there exists a string t such that $C(s, t) = yes$.

Yes-instance

"certificate" or "witness"

Def. **NP** is the set of problems for which there exists a poly-time certifier.

- $C(s, t)$ is a poly-time algorithm.
- Certificate t is of polynomial size: $|t| \leq p(|s|)$ for some polynomial $p(\cdot)$.

Remark. **NP** stands for nondeterministic polynomial time.

6

Decision problem / language

$L \subseteq \{0,1\}^*$ (e.g., all prime numbers)

Algorithm for L:

Given x, decide if $x \in L$.

Usual alg: deterministic.

usually, deterministic (randomized) algos for L.

Non-deterministic algos:

(1) $L \in NP$ if L has a "guess & check" algo:

Given $x \in \{0,1\}^n$,
(*) guess string $y \in \{0,1\}^{poly(n)}$
& check that (x,y) satisfy some condition. $\in P$

(*) non-det. guess

<u>Ex:</u> Given $x \in \{0,1\}^n$, decide if x is composite?

Composite $\in NP$:
"Guess & check" algo:
Given x, <u>guess</u> y = divisor of x

Given x, guess y = divisor of x

Check that y divides x.

If y passes the check, Accept, else Reject.

$$\text{Prover} \xleftarrow{\quad} x \xrightarrow{\quad} \text{Verifier}$$

$$\text{Prover} \xrightarrow{\quad y \quad} \text{Verifier}$$

$V(x,y)$

V is det. poly time

$x \in L \Rightarrow \exists y, |y| \le poly(|x|)$ s.t. $V(x,y) = 1$

$x \notin L \Rightarrow \forall y, |y| \le |x|^{O(1)}$ $V(x,y) = 0$

Certifiers and certificates: composite

COMPOSITES. Given an integer s, is s composite?

Certificate. A nontrivial factor t of s. Such a certificate exists iff s is composite. Moreover $|t| \le |s|$.

Certifier. Check that $1 < t < s$ and that s is a multiple of t.

instance s	437669
certificate t	541 or 809

← 437,669 = 541 × 809

Conclusion. COMPOSITES $\in$ **NP**. $\longleftarrow$ in fact, COMPOSITES $\in$ **P**

7

Certifiers and certificates: satisfiability

SAT. Given a CNF formula Φ, does it have a satisfying truth assignment?
3-SAT. SAT where each clause contains exactly 3 literals.

Certificate. An assignment of truth values to the Boolean variables.

Certifier. Check that each clause in Φ has at least one true literal.

instance s $\Phi = \left(\overline{x_1} \vee x_2 \vee x_3\right) \wedge \left(x_1 \vee \overline{x_2} \vee x_3\right) \wedge \left(\overline{x_1} \vee x_2 \vee x_4\right)$

certificate t $x_1 = \mathit{true},\ x_2 = \mathit{true},\ x_3 = \mathit{false},\ x_4 = \mathit{false}$

2-SAT $\in$ P $(x \vee y) \wedge (\bar{z} \vee x) \wedge \ldots$

Conclusions. SAT $\in$ **NP**, 3-SAT $\in$ **NP**.

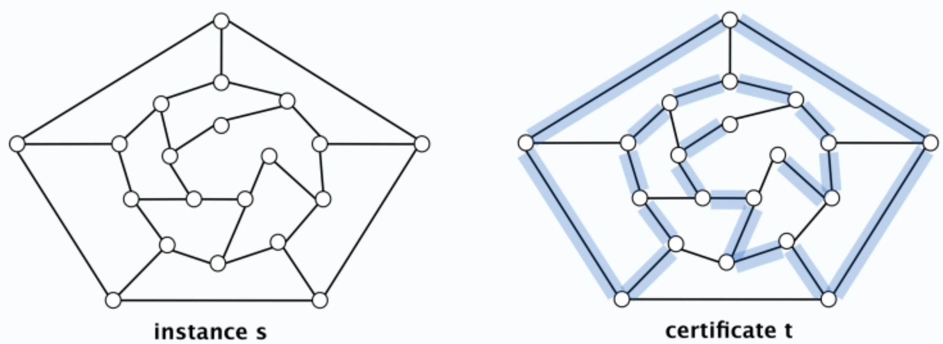
8

Certifiers and certificates: Hamilton path

HAM-PATH. Given an undirected graph $G = (V, E)$, does there exist a simple path P that visits every node?

Certificate. A permutation of the n nodes.

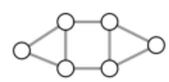
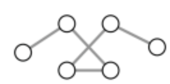
Certifier. Check that the permutation contains each node in V exactly once, and that there is an edge between each pair of adjacent nodes.



Conclusion. HAM-PATH $\in$ **NP**.

Definition of NP

NP. Decision problems for which there is a poly-time certifier.

Problem	Description	Algorithm	yes	no
L-SOLVE ∈P	Is there a vector x that satisfies $Ax = b$?	Gauss–Edmonds elimination	$\begin{bmatrix}0&1&1\\2&4&-2\\0&3&15\end{bmatrix}, \begin{bmatrix}4\\2\\36\end{bmatrix}$	$\begin{bmatrix}1&0&0\\1&1&1\\0&1&1\end{bmatrix}, \begin{bmatrix}1\\1\\1\end{bmatrix}$
COMPOSITES ∈NP	Is x composite ?	AKS (2002)	51	53
FACTOR	Does x have a nontrivial factor less than y ?	?	(56159, 50)	(55687, 50)
SAT	Given a CNF formula, does it have a satisfying truth assignment?	?	$\neg x_1 \lor x_2 \lor \neg x_3$ $x_1 \lor \neg x_2 \lor x_3$ $\neg x_1 \lor \neg x_2 \lor x_3$	$\neg x_2$ $x_1 \lor x_2$ $\neg x_1 \lor x_2$
3-COLOR	Can the nodes of a graph G be colored with 3 colors?	?		
HAM-PATH	Is there a simple path between u and v that visits every node?	?		

$L \in P \Rightarrow NP = P$

P NP P ⊆ NP ⇒ P ≟ NP

? $1,000,000

10

Definition of NP

NP. Decision problems for which there is a poly-time certifier.

> " *NP captures vast domains of computational, scientific, and mathematical endeavors, and seems to roughly delimit what mathematicians and scientists have been aspiring to compute feasibly.* " — Christos Papadimitriou

> " *In an ideal world it would be renamed P vs VP.* " — Clyde Kruskal

" In an ideal world it would be renamed P vs VP. " — *Clyde Kruskal*

11

P, NP, and EXP

P. Decision problems for which there is a poly-time algorithm.

NP. Decision problems for which there is a poly-time certifier.

EXP. Decision problems for which there is an exponential-time algorithm.

Claim. **P** $\subseteq$ **NP**.

Pf. Consider any problem $X \in$ **P**.

- By definition, there exists a poly-time algorithm $A(s)$ that solves X.
- Certificate $t = \varepsilon$, certifier $C(s, t) = A(s)$. ▪

Claim. **NP** $\subseteq$ **EXP**.

Pf. Consider any problem $X \in$ **NP**.

- By definition, there exists a poly-time certifier $C(s, t)$ for X, where certificate t satisfies $|t| \leq p(|s|)$ for some polynomial $p(\cdot)$.
- To solve input s, run $C(s, t)$ on all strings t with $|t| \leq p(|s|)$.
- Return *yes* if $C(s, t)$ returns *yes* for any of these potential certificates. ▪

Can we do better?

NP $\subseteq$ P ??

Remark. Time-hierarchy theorem implies **P** $\subsetneq$ **EXP**.

12

The main question: P vs. NP

Q. How to solve an instance of 3-SAT with n variables?

A. Exhaustive search: try all 2^n truth assignments.

Q. Can we do anything substantially more clever?

Conjecture. $\underbrace{\text{No poly-time algorithm}}_{\text{"intractable"}}$ for 3-SAT.

$2^{\frac{n}{10}}$???

The main question: P vs. NP

Does P = NP? [Cook 1971, Edmonds, Levin, Yablonski, Gödel]

Is the decision problem as easy as the certification problem?

not NP complete

If yes. Efficient algorithms for 3-SAT, TSP, 3-COLOR, FACTOR, ...

If no. No efficient algorithms possible for 3-SAT, TSP, 3-COLOR, ...

Consensus opinion. Probably no.

Possible outcomes

P ≠ NP.

> *" I conjecture that there is no good algorithm for the traveling salesman problem. My reasons are the same as for any mathematical conjecture: (i) It is a legitimate mathematical possibility and (ii) I do not know."*
>
> *— Jack Edmonds 1966*

Possible outcomes

P ≠ NP.

" In my view, there is no way to even make intelligent guesses about the answer to any of these questions. If I had to bet now, I would bet that P is not equal to NP. I estimate the half-life of this problem at 25–50 more years, but I wouldn't bet on it being solved before 2100. "

— Bob Tarjan (2002)

" We seem to be missing even the most basic understanding of the nature of its difficulty.... All approaches tried so far probably (in some cases, provably) have failed. In this sense P =NP is different from many other major mathematical problems on which a gradual progress was being constantly done (sometimes for centuries) whereupon they yielded, either completely or partially. "

— Alexander Razborov (2002)

Possible outcomes

P = NP.

> “ *I think that in this respect I am on the loony fringe of the mathematical community: I think (not too strongly!) that P=NP and this will be proved within twenty years. Some years ago, Charles Read and I worked on it quite bit, and we even had a celebratory dinner in a good restaurant before we found an absolutely fatal mistake.* ”
>
> — *Béla Bollobás (2002)*

Other possible outcomes

P = **NP**, but only $\Omega(n^{100})$ algorithm for 3-SAT.

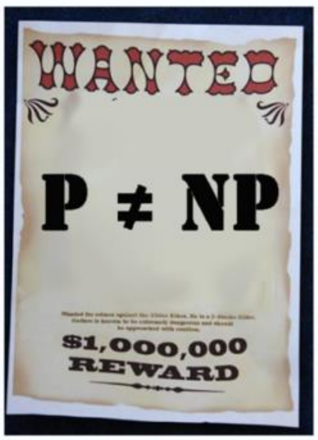
P ≠ **NP**, but with $O(n^{\log^* n})$ algorithm for 3-SAT.

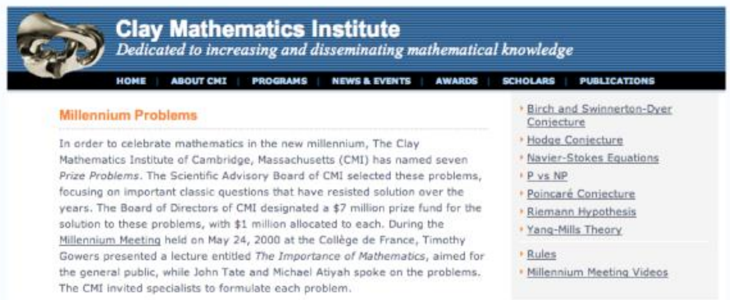
P = **NP** is independent (of ZFC axiomatic set theory).

> *" It will be solved by either 2048 or 4096. I am currently somewhat pessimistic. The outcome will be the truly worst case scenario: namely that someone will prove "P = NP because there are only finitely many obstructions to the opposite hypothesis"; hence there exists a polynomial time solution to SAT but we will never know its complexity! "* — Donald Knuth

Millennium prize

Millennium prize. $1 million for resolution of **P = NP** problem.





Looking for a job?

Some writers for the Simpsons and Futurama.

- J. Steward Burns. *M.S. in mathematics* (*Berkeley '93*).
- David X. Cohen. *M.S. in computer science* (*Berkeley '92*).
- Al Jean. *B.S. in mathematics*. (*Harvard '81*).
- Ken Keeler. *Ph.D. in applied mathematics* (*Harvard '90*).
- Jeff Westbrook. *Ph.D. in computer science* (*Princeton '89*).

Copyright © 1990, Matt Groening

Copyright © 2000, Twentieth Century Fox

20

Polynomial transformation

Def. Problem X polynomial (Cook) reduces to problem Y if arbitrary instances of problem X can be solved using:

- Polynomial number of standard computational steps, plus
- Polynomial number of calls to oracle that solves problem Y.

Def. Problem X polynomial (Karp) transforms to problem Y if given any input x to X, we can construct an input y such that x is a *yes* instance of X iff y is a *yes* instance of Y.

we require $|y|$ to be of size polynomial in $|x|$

Note. Polynomial transformation is polynomial reduction with just one call to oracle for Y, exactly at the end of the algorithm for X. Almost all previous reductions were of this form.

Open question. Are these two concepts the same with respect to **NP**?

we abuse notation $\leq_p$ and blur distinction

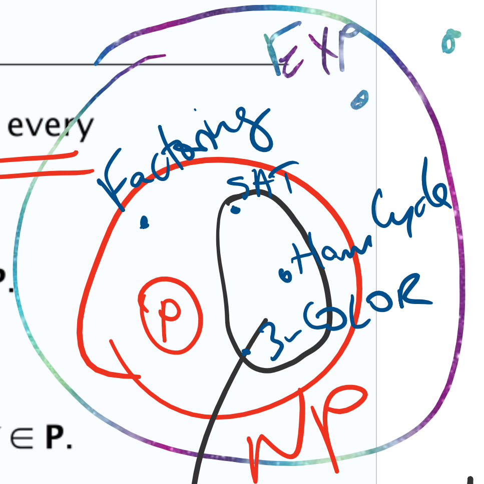
24

NP-complete

(1)

NP-complete. A problem $Y \in \mathbf{NP}$ with the property that for every problem $X \in \mathbf{NP}$, $X \leq_p Y$.

X is reducible to Y



Theorem. Suppose $Y \in \mathbf{NP}$-complete. Then $Y \in \mathbf{P}$ iff $\mathbf{P} = \mathbf{NP}$.

Pf. $\Leftarrow$ If $\mathbf{P} = \mathbf{NP}$, then $Y \in \mathbf{P}$ because $Y \in \mathbf{NP}$.

Pf. $\Rightarrow$ Suppose $Y \in \mathbf{P}$.

- Consider any problem $X \in \mathbf{NP}$. Since $X \leq_p Y$, we have $X \in \mathbf{P}$.
- This implies $\mathbf{NP} \subseteq \mathbf{P}$.
- We already know $\mathbf{P} \subseteq \mathbf{NP}$. Thus $\mathbf{P} = \mathbf{NP}$. ▪

NP-complete "hardest" in NP

Fundamental question. Do there exist "natural" **NP**-complete problems?

Yes. Lots!!! SAT, HamCycle, 3-COLOR ....