Tuesday, April 19, 2022 10:59 AM

Last Time: How to prove that a problem is hard?

* P = class of all problems w/ poly-time algo.

* $L_1 \le_p L_2$: poly-time reduction

  Fact 1: $L_2 \in P \Rightarrow L_1 \in P$

  equiv $L_1 \notin P \Rightarrow L_2 \notin P$

  Fact 2: $L_1 \le_p L_2 \le_p L_3 \Rightarrow L_1 \le_p L_3$

Q: How to get the first hard problem?

---

* NP: Nondeterministic Polynomial

Intuition:

Det. Brute force: Enumerate & check.

e.g. VC: enumerate all subsets S of vertices (2^n many)

& for each check if $|S| \le k$ & if S is a VC.

Non-det. Poly-time: "guess" a subset S & check if $|S| \le k$ & S is a VC.

guess-n-check.

Def: NP = class of all decision problems of the form.

Input: x

Output: YES iff $\exists$ y (guess) s.t. $c(x, y)$ (check) is true.

[illegible] time / verifies.

Input: x ...

Output: YES iff $\exists\, y$ s.t. $C(x,y)$ ...

y ← certificate; C ← certifier / verifier.

where ① y is poly size in $|x|$.

② C is poly-time algo.

Ex: Independent set $\in NP$.

Pf: Given input to IS: $G=(V,E)$, int. K.

① certificate: $S \subseteq V$ ← poly size $|S| \le |V|$

② Certifier: $|S| \ge K$, and $\forall u,v \in S \Rightarrow uv \notin E$ ← Poly-time.

⋮

Set cover: $\in NP$

Travelling Salesman Problem $\in NP$.

⋮

Facts: $P \subseteq NP \subseteq EXP$ ← class of problems that have exp-time algo. $O(2^{p(n)})$ time, $p(n)$ is a polynomial in n.

Pf: $(P \subseteq NP)$: Certifier will ignore the certificate & solve the problem

$(NP \subseteq EXP)$: $|y| \le n^c$ where $n = |x|$

\# possible $y = 2^{n^c}$

Try all certificates.

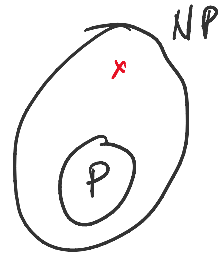
"Million-Dollar" Conjecture:

... NP

NP

"Million-Dollar ..."

$P \neq NP$



idea: to find a hard problem in NP
take the hardest problem in NP.

Def: L is NP-complete iff

① $L \in NP$

$\boxed{② \ \forall L' \in NP, \ L' \leq_p L}$ ← $\underline{\text{NP-hard.}}$

Fact 4: Let L be NP-complete

$L \notin P \iff P \neq NP$.

Pf: ($\Rightarrow$) suppose $L \notin P$. But $L \in NP \Rightarrow P \neq NP$

($\Leftarrow$) suppose $\underline{L \in P}$ then by NP-hardness of L

$\forall L' \in NP, \ L' \leq_p L \Rightarrow L' \in P$ ($\because$ Fact 1)

$\Rightarrow NP \subseteq P$. And by Fact 3 $P \subseteq NP$.

$\Rightarrow NP = P$. ∎

World "Assuming" $P \neq NP$

"First" hard problem in NP: Satisfiability (SAT)

"First" hard problem in NP: Satisfiability

Input: Boolean formula in n var

$F(x_1, \ldots, x_n)$

Output: YES iff $\exists$ assignment A of Boolean values to the variables

s.t. F evaluates to true on A.

eg. $F(x_1, x_2, x_3) = (\bar{x}_1 \vee \overline{x_2 \wedge x_3}) \wedge (x_1 \vee x_2)$

YES: $(x_1 = 0, x_2 = 1, x_3 = 0 \text{ or } 1) = A$

Cook-Levin Theorem (1971): SAT is NP-Complete.

pf sketch: ① SAT $\in$ NP (exe).

② SAT is NP-hard:

Need to give poly-time reduction from any $L \in NP$ to SAT!

Say L is in NP:

Input: x of L.

Output: YES iff $\exists y$ s.t. $C(x,y)$ is true

Poly-time Alg'm/

SAT. $\longleftarrow$ TM M.

Intuition. Recall: TM is just a "finite state" machine with read + write tape.

$x[i, j]$ = content of tape cell j

Intu.

Create var $x[i, j]$ = content of tape cell j in step i.

- # variables is poly.
- Given value of $x[i, j]$ we know exactly know value of $x[i+1, j']$ $\forall j'$.

Capture it through a boolean formula