Wednesday, April 18, 2018 2:14 PM

NP- Completeness

Q: When is a problem in NP ?

NP = { languages which admit a Non-deterministic poly time Turing machine }

NP is the set of all languages L s.t

$\exists$ poly-time verifier V

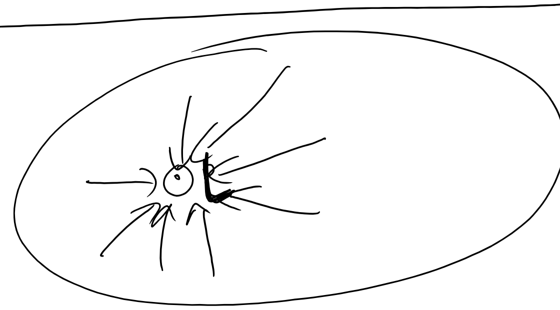
$\forall x \in L$, $\exists$ "proof" $\pi(x)$ s.t $V(x, \pi(x)) = 1$

where V's runtime & $|\pi(x)|$ should be polynomial in $|x|$

$\forall x \notin L$, $\forall$ proofs $\pi(x)$, $V(x, \pi(x)) = 0$

NP- Completeness

L is NP-complete if every problem $L' \in NP$ is poly-time reducible to L



---

3 Coloring $\in$ NP

Given a graph G, tell if its 3-colourable or not?

Corresponding language L = { 3-colourable graphs }

Given graph x, tell if $x \in L$ or not?

If $x \in L$, prover supplies a 3-coloring

| | | | | | | |

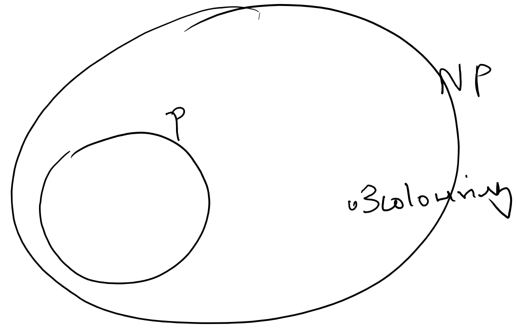
n values from {0, 1, 2}

Verifier enumerates all edges & checks that the colours supplied by prover are different

If $x \notin L_1$, no proof can pass the verifier !!

Given a problem L, how to quantify that L is hard?

e.g.: Is 3-colouring really a "hard" problem?



Way to show :-

$\boxed{\text{Prove that}}$ 3-colouring is NP-hard (ie) every problem in NP can be reduced in poly-time to 3-colouring

---

Q| How to show that 3-colouring is NP hard?

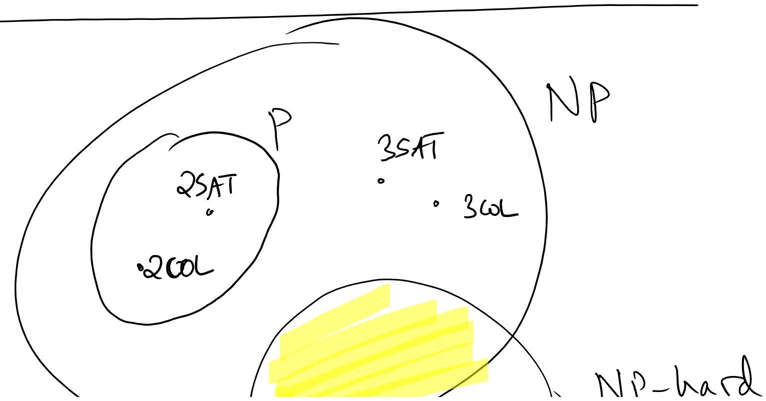
$$L_1 \xrightarrow{f} L_2$$

~~$\forall x$~~ given x, output $f(x)$ & f runs in poly-time

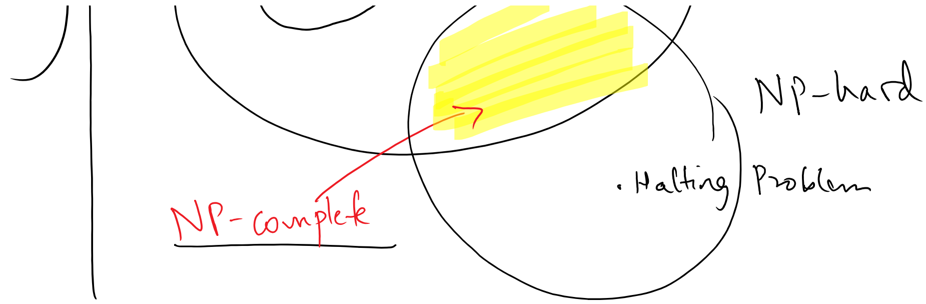
If $x \in L_1 \rightleftarrows f(x) \in L_2$

f along with a poly-time algo for L_2 $\Rightarrow$ polytime algo for L_1

To show that a problem in NP is unlikely to be in P, show that it is also NP-hard



show that ... NP-hard



Cook-Levin theorem :-

(★) 3SAT is NP-complete (ie) it is in NP & NP-hard

↓ $x_1, x_2 \dots x_n$

$C_1, C_2 \dots C_m$

$C_i = \bar{x}_{i_1} \vee x_{i_2} \vee x_{i_3}$

Is there an assignment to $x_1 \dots x_n$ st all clauses are satisfied ??

OK, great!
but what about 3COL ??

$NP \rightarrow 3SAT \xrightarrow{f} 3COL$

Such a reduction will show that 3COL is NP-hard !!

$\Rightarrow$ Unlikely that 3COL has a poly-time algorithm

Using poly-time reductions, we could show NP-completeness but were stuck on showing hardness of approximation.

Example MAJOR OPEN QN. UNTIL 90s.

→ Does ~~Not~~ Max 3SAT have a PTAS ?

→ Does Max Independent set have a PTAS?

→ Does Min vertex cover have a PTAS ?

... 3SAT can be solved in poly-time,

If Max 3SAT can be solved in poly-time, so can 3SAT

Then, Does it have a PTAS? (ie)

$\forall$ constant $\varepsilon > 0$, can we get a $(1-\varepsilon)$-approx to Max 3SAT in time poly in n??

Similarly is there a $(1+\varepsilon)$-approximation to vertex cover in poly(n) time?

Can't hope to prove these results by reductions like standard NP-completeness reductions

---

Until 92, when [ALMSS '92] showed that there is some constant $\varepsilon_0 > 0$ st it is NP-hard to get a $(1-\varepsilon_0)$-approximation to Max 3SAT

$\Downarrow$

PTAS is ruled out!!

$\boxed{\text{PCP-theorem}}$

Surprisingly

Goal of PCPs was originally not to show hardness of approximation, but rather to obtain a better understanding of NP!!

PCP is a complexity class introduced to study the power of the verifier?

(ie) Can I restrict the verifier & if so, by how much??

(ie) can $\perp$

so, by how much ??

For example, can we restrict the usual verifier in NP to only query 5 bits of the proof & decide his accept / reject based on that ??

Seems like the verifier is too restrictive !!

Allow the verifier some randomness !!

PCP $[r(n), q(n)]$ is a class of languages L st

$\exists$ Verifier (poly-time, has access to $r(n)$ - random coins) st

$\forall x \in L$, $\exists$ proof $\pi(x)$ st Verifier probes only $q(n)$ bits of the proof, & accepts w.p $\boxed{1}$

$\forall x \notin L$, $\forall$ proofs $\pi(x)$, Verifier rejects w.p $\geq \frac{1}{2}$.

PCP theorem [restated]

$$PCP[O(\log n), 3] = NP$$