**NP-Completeness**

A. What is a problem in NP?

NP = \{ language which admit a non-deterministic polytime Turing machine \}

NP is the set of all languages L s.t.

\[ \exists \text{ poly-time verifier } V \]

\[ \forall x \in L, \exists \text{ "proof" } \Pi(x) \text{ s.t. } V(x, \Pi(x)) = 1 \]

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

\[ \forall x \in L, \forall \text{ proofs } \Pi(x), \forall V(x, \Pi(x)) = 0 \]

---

**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 it's 3-colorable or not?

Corresponding language L = \{ 3-colorable graphs \}

Given graph G, tell if \( x \in L \) or not?

If \( x \in L \), prove supplies a 3-coloring of \( x \) where values from \{0, 1, 2\}
Very few enumerates all edges & checks that the colour supplied by prov is different.

If \( x \notin L \), 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:

**Prove that 3-colouring is NP-hard (i.e.)**

Every problem in NP can be reduced in poly-time to 3-colouring.

Q1: How to show that 3-colouring is NP-hard?

\[ L_1 \xrightarrow{f} L_2 \]

\( f \) given \( x \), output \( f(x) \) & \( f \) runs in poly-time.

If \( x \in L_1 \Rightarrow f(x) \in L_2 \)

\( f \) along with a poly-time algo for \( L_2 \)\n
\( \Rightarrow \) 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.
Cook-Levin Theorem:

\(3\text{SAT}\) is \(NP\)-complete (i.e., \(NP \leq \text{NP-complete}\)).

Is there an assignment to \(x_1, x_2, \ldots, x_n\) such that all clauses are satisfied?

\[ c_i = \overline{x}_i \lor x_{i+1} \lor \overline{x}_{i+2} \]

\(3\text{SAT} \leq_{\text{poly}} 3\text{COL}\)

Such a reduction will show that \(3\text{COL}\) is \(NP\)-hard!!

\(\Rightarrow\) Unlikely that \(3\text{COL}\) 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 ON until 90°.

\[ \rightarrow \text{Does Max 3-SAT have a PTAS?} \]

\[ \rightarrow \text{Does Max Independent set have a PTAS?} \]

\[ \rightarrow \text{Does Min vertex cover have a PTAS?} \]

...Can be solved in poly-time...
If Max 3SAT can be solved in poly-time, so can 2SAT.

Then, does it have a PTAS? (i.e.)

+ 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 $\varepsilon > 0$ s.t.

it is NP-hard to get a $(1-\varepsilon)$-approximation to Max 3SAT.

Surprisingly, PCP theorem PTAS is ruled out!

Goal of PCPs was originally not to throw boundaries of approximation, but rather to obtain a better understanding of NP II.

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

(i.e.) Can I restrict the verifier & if so, by how much??
(10) Can I log how much ??

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

Seems like the verifier is too restrictive !!

Lot the verifier some randomness !!

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

\[ 3 \text{ Verifier (poly-time, has access to } r(n) \text{-random) st} \]

\[ \forall x \in L, \exists \text{ proof } \pi(x) \text{ st Verifier probes only } q(n) \text{ bits of the proof, & accepts } w.p. \ \frac{1}{2} \]

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

$PCP$ theorem [restated]

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