Lecture 25 Complexity Classes P and NP

All the algorithms that we studied so far were polynomial time algorithms. On inputs of size n, the running time was $O(n^k)$ for some constant k. Let us formalize this.

A computational task faced by a computer is modelled as a language recognition problem. In the first place, let us restrict our problems to "decision problems". These are problems with yes/no answers.

For example, given a graph G, is it connected? The output is yes/no.

We associate a language $\mathcal{L}_{conn}$ with this decision problem: $\mathcal{L}_{conn} = \{G : G \text{ is connected}\}$.

Given any language $\mathcal{L}$ and an input string x, the corresponding language recognition problem is: does $x \in \mathcal{L}$?

Algorithm A solves this problem if

$$x \in \mathcal{L} \Rightarrow A \text{ says "yes"}$$
$$x \notin \mathcal{L} \Rightarrow A \text{ says "no"}$$

Does every language $\mathcal{L}$ have an algorithm that recognizes it? NO - too many languages compared to algorithms

(Please justify this)

Consider a language $\mathcal{L}$ that has an algorithm A that recognizes it. We want to study the performance of A.

Let $t_A(n)$ = max. time taken by A on inputs of length n.

$P = \{ \mathcal{L} : \exists$ an algorithm A that recognizes $\mathcal{L}$ such that $t_A(n) = O(n^k)$ for some constant $k \}$

Ex. $\mathcal{L}_{conn}$, $\mathcal{L}_{2COL}$, $\mathcal{L}_{primes}$

$$w \longrightarrow \boxed{\begin{array}{c} A \text{ runs in} \\ O(|w|^k) \text{ time} \end{array}} \begin{array}{l} \longrightarrow \text{yes if } w \in \mathcal{L} \\ \longrightarrow \text{no if } w \notin \mathcal{L} \end{array}$$

P is the set of languages that have efficient algorithms to recognize them.

There are many natural decision problems that are not known to be in P.

Ex. 1) IND-SET: given a graph G and an integer k, is there an independent set of size k in G?

[A subset S of vertices is an independent set if there is no edge between any 2 vertices in S.]

2) 3SAT: given a formula ϕ in 3CNF, is ϕ satisfiable?

3) 3COL: given a graph G, is G 3-colourable?

We do not know efficient algorithms to recognize the above languages. But can we enhance our computational model so that these problems can be efficiently solved?

What if our computers can ~~the~~ make guesses or non-deterministic choices? It turns out that then these languages have efficient algorithms to recognize them.

What is non-determinism?

A non-deterministic poly. time algo. for 3SAT

Input: a Boolean formula ϕ

Step 1: Guess a true/false assignment for the variables in ϕ.

Step 2: Check if the above assignment satisfies ϕ. If so then say yes else say no.

Note that Step 2 is entirely deterministic. It is not clear how such an algorithm proceeds. It depends on what assignment is guessed in Step 1 and for different assignments, the final conclusion may be different.

It is possible for the algorithm to say "no" even if ϕ is satisfiable. However if ϕ is not satisfiable then the algo. ~~never~~ says "yes". This is the essence of non-deterministic computation. A non-det. algo. A recognizes lang. $\mathcal{L}$ if $x \in \mathcal{L} \Rightarrow \exists$ a sequence of guesses made by A that makes A say yes

$x \notin \mathcal{L} \Rightarrow$ A always says no.

Another non-deterministic algo. (for independent set)

Input: a graph G & a number k

Step 1: Guess a subset S of V of size k

Step 2: Return yes if S forms an independent set else return no.

Another non-deterministic algo (for 3COL)

Input: a graph $G = (V, E)$

Step 1: Guess a partition (V_1, V_2, V_3) of V.

<u>Step 2</u>: Return yes if each V_i (for $i = 1,2,3$) is an independent set

else return no.

$$T_A(n) = \max_{w: |w|=n} \; \max_{\text{guess seq. } y} \; \text{time taken by } A \text{ on input } w \text{ \& guess } y$$

NP = { $\mathcal{L}$: $\mathcal{L}$ is recognized by a non-det. algo. A whose running time $T_A(n)$ is $O(n^k)$ for some constant k }

<u>Non-det. polynomial time algorithm A</u>

1. Make a guess y (the length of the guess is polynomial in input size)
2. Check/verify the guess.
   - this is a deterministic poly. time algo. that works with the input w & guess y.

* If $\phi \in$ 3SAT then $\exists$ an assignment that makes Step 2 say yes.
* If $G \in$ 3COL then $\exists$ a colouring that makes Step 2 say yes.
* If $(G, k) \in$ IND-Set then $\exists$ a subset of V that makes Step 2 say yes.

<u>$\mathcal{L} \in$ NP</u>: For $w \in \mathcal{L}$, there are "short" (of size poly in $|w|$) witnesses to w's membership in $\mathcal{L}$.

For $w \notin \mathcal{L}$, are there "short" witnesses to w's non-membership in $\mathcal{L}$? — not that we know of.

$\phi \notin$ 3SAT → is there a short guess for this?
$G \notin$ 3COL → is there a short guess for this?
$(G, k) \notin$ IND-Set → is there a short guess for this?

So the languages $\overline{\mathcal{L}}_{SAT} = \{\phi : \phi \text{ is not satisfiable}\}$,

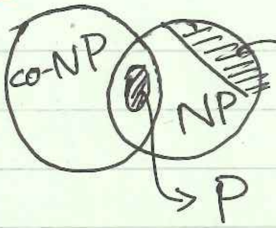
$\overline{\mathcal{L}}_{3COL} = \{G : G \text{ is not 3-colourable}\}$,

$\overline{\mathcal{L}}_{IND\text{-}Set} = \{(G, k) : G \text{ has no independent set of size } k\}$

are not known to be in NP.

The complexity class co-NP $= \{\overline{\mathcal{L}} : \mathcal{L} \in NP\}$.

Exercise: Show that $P \subseteq NP \cap \text{co-}NP$.



the hardest problems in the class NP.

Reductions: This makes precise what it means for a problem to be at least as hard as another. A language $\mathcal{L}_1$ is polynomial time reducible to language $\mathcal{L}_2$, denoted by $\mathcal{L}_1 \leq_P \mathcal{L}_2$ if there exists a polynomial time computable function f such that for each $x \in \Sigma^*$

$$x \in \mathcal{L}_1 \iff f(x) \in \mathcal{L}_2$$

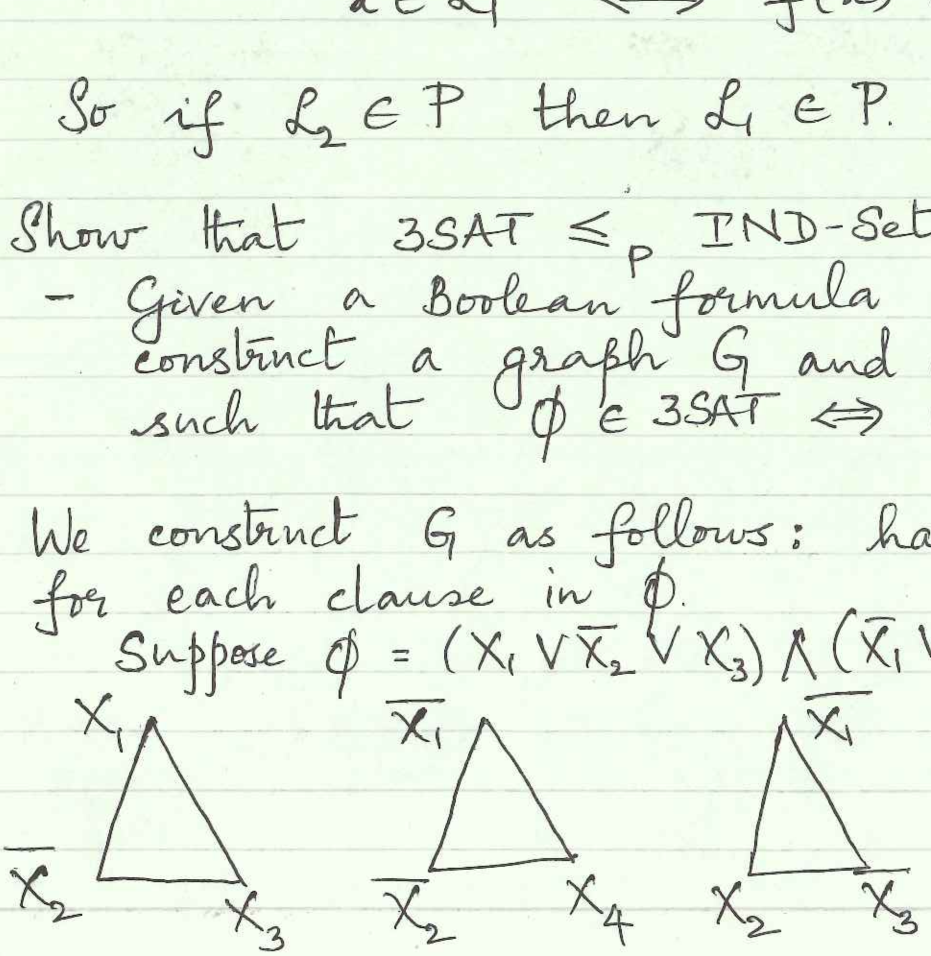
So if $\mathcal{L}_2 \in P$ then $\mathcal{L}_1 \in P$.

Show that $3SAT \leq_P$ IND-Set.

- Given a Boolean formula ϕ in 3CNF, (n variables, m clauses) construct a graph G and an integer k such that $\phi \in 3SAT \iff (G, m) \in$ IND-Set.

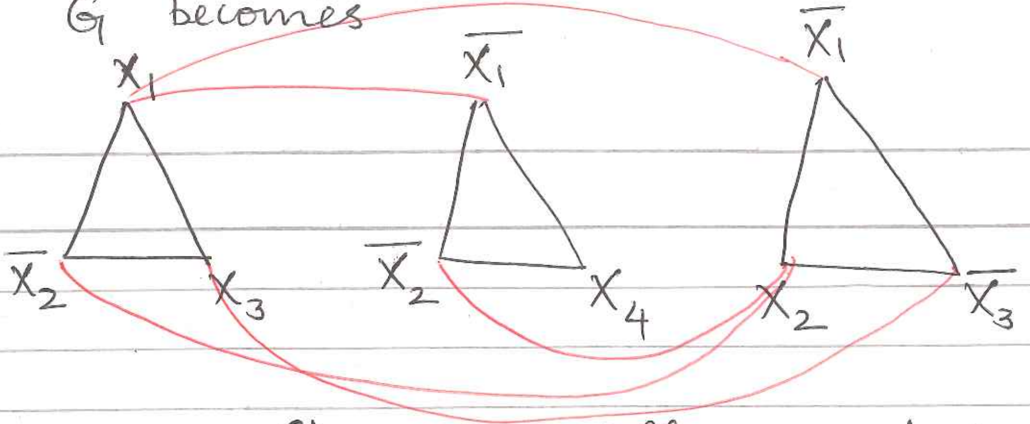
We construct G as follows: have a triangle for each clause in ϕ.

Suppose $\phi = (X_1 \vee \overline{X}_2 \vee X_3) \wedge (\overline{X}_1 \vee \overline{X}_2 \vee X_4) \wedge (\overline{X}_1 \vee X_2 \vee \overline{X}_3)$



- Make every X_i and $\overline{X}_i$ adjacent for $i = 1, 2, 3, \ldots$

So G becomes



Exercise. Show the following claims:

- if ϕ is satisfiable then G has an independent set of size m.
- if G has an independent set of size m then ϕ is satisfiable.

If for every language $\mathcal{L}$ in NP we have $\mathcal{L} \leq_p \mathcal{L}'$ then $\mathcal{L}'$ is called NP-hard.
If in addition $\mathcal{L}' \in NP$ then $\mathcal{L}'$ is NP-complete. Do NP-complete languages exist?

Cook's theorem. 3SAT is NP-complete.

1) It follows from our reduction that IND-Set is NP-complete.

2) Clique is NP-complete. (Please show this.)

3) Vertex-Cover is NP-complete.
Vertex-Cover $= \{(G, k) :$ G has a vertex cover of size $k\}$.
Recall that $C \subseteq V$ is a vertex cover if for every edge (u, v) in G, either u or v is in C.

Exercise. Show that $(G, k) \in$ IND-Set $\iff (G, n-k) \in$ Vertex-Cover.
So IND-Set $\leq_p$ Vertex-Cover.