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 attention 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 L_{conn} with this decision problem: $L_{conn} = \{G : G \text{ is connected}\}$.

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

Algorithm A solves this problem if

$x \in L \Rightarrow$ A says "yes"

$x \notin L \Rightarrow$ A says "no"

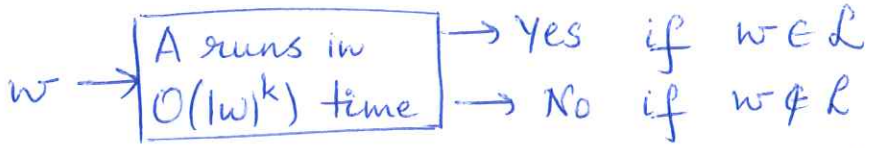
Does every language L have an algorithm that recognizes it? NO - too many languages compared to algorithms (Please justify this.)

Consider a language L that has an algorithm A that recognizes it. We want to study the performance of A.

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

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

Ex. L_{conn}, L_{2COL}, L_{Primes}



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 the vertex set 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 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 algorithm 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 is different.

It is possible for the algorithm to say "no" even if ϕ is satisfiable. However if ϕ is not satisfiable then the algorithm never says "yes". This is the essence of non-deterministic computation.

A non-deterministic algorithm A recognizes language $\mathcal{L}$ if $x \in \mathcal{L} \Rightarrow \exists$ a guess that makes A say "yes"

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

A non-deterministic algo. for independent set

Input: A graph G and 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".

A non-deterministic algo. for 3COL

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

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

Step 2: Return "yes" if each V_i (for $i = 1, 2, 3$) is an independent set; else return "no".

Let $T_A(n) = \max_{w : |w| = n} \max_{\text{guess } y}$ Time taken by A on input w & guess y

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

Non-deterministic polynomial time algorithm A

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

- If $\phi \in$ 3SAT then $\exists$ an assignment of true/false values 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".

$L \in NP$: For $w \in L$, there are "short" (of size poly($|w|$)) witnesses to w's membership in L.

For $w \notin L$, are there "short" witnesses to w's non-membership in L?

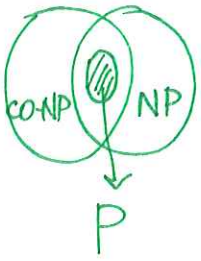
$\phi \notin$ 3SAT $\rightarrow$ Is there a short guess for this?

$G \notin$ 3COL $\rightarrow$ Is there a short guess for this?

$(G,k) \notin$ IND-Set $\rightarrow$ Is there a short guess for this?

The languages $L_{3SAT} = \{ \phi : \phi$ is not satisfiable$\}$,
$L_{3COL} = \{ G : G$ is not 3-colourable$\}$,
$L_{IND\text{-}Set} = \{ (G,k) : G$ has no independent set of size $k \}$
are not known to be in NP.

The complexity class co-NP $= \{\bar{L} : L \in NP\}$.
Show that $P \subseteq NP \cap co\text{-}NP$. (Exercise.)



Reductions: This makes precise what it means for a problem to be at least as hard as another.

A language L_1 is polynomial time reducible to language L_2, denoted by $L_1 \leq_P L_2$, if there exists a polynomial time computable fn. f such that for each $x \in \Sigma^*$: $x \in L_1 \iff f(x) \in L_2$.

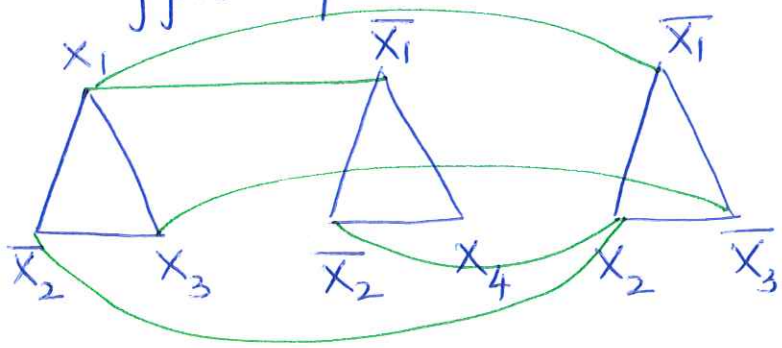
So $L_2 \in P \Rightarrow L_1 \in P$

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 $\iff (G, k) \in$ IND-Set.

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

Suppose $\phi = (x_1 \vee \bar{x}_2 \vee x_3) \wedge (\bar{x}_1 \vee \bar{x}_2 \vee x_4) \wedge (\bar{x}_1 \vee x_2 \vee \bar{x}_3)$



We'll make every x_i & $\bar{x}_i$ adjacent for $i = 1, 2, \ldots$

Exercise: Show the following claims.

(1) If ϕ is satisfiable then G has an independent set of size m.

(2) If G has an independent set of size m then ϕ is satisfiable.

If for every language L in NP we have $L \leq_p L'$ then L' is called NP-hard.

If in addition $L' \in NP$ then L' is called NP-complete. Do NP-complete languages exist?

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

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

2) Clique is NP-complete. (Exercise.)

3) Vertex-Cover is NP-complete.

Vertex-Cover = $\{(G,k)$: G has a vertex cover of size $k\}$.

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

So IND-Set $\leq_p$ Vertex-Cover.

4) 3COL is NP-complete.

We will use a variant of SAT called NAE-SAT (not all equal SAT) to show the NP-hardness of 3COL. In NAE-SAT we want to know if there is a true/false assignment to the input formula so that:

- in every clause there is at least 1 true literal and 1 false literal.

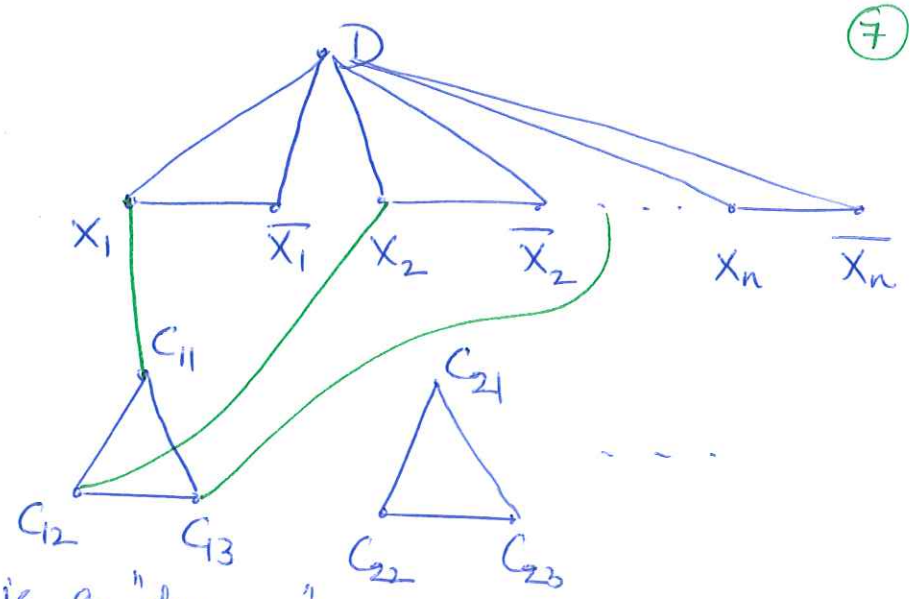
NAE-SAT is also NP-complete.

We will show that NAE-SAT $\leq_p$ 3COL.

Given a 3CNF formula ϕ that is input to NAE-SAT, we will build a graph G_ϕ so that

$\phi \in$ NAE-SAT $\iff G_\phi \in$ 3COL.

The graph G_ϕ



The top vertex D is a "dummy".
$x_1, \ldots, x_n$ are the n variables in ϕ.
Suppose the first clause is $x_1 \vee x_2 \vee \bar{x}_3$.

Claim 1. If $\phi \in$ NAE-SAT then $G_\phi \in$ 3COL.

Colour the vertex D green. Assign red colour to all true literals and blue to all false literals in the top layer.

The vertices in the triangles in the lower layer can be coloured red/blue/green appropriately so that the resulting colouring is valid. (Please check this.)

Claim 2. If $G_\phi \in$ 3COL then $\phi \in$ NAE-SAT.

In this 3-colouring of G_ϕ, assume (without loss of generality) that the vertex D is coloured green. So among $x_i, \bar{x}_i$ (for $1 \le i \le n$), one is coloured red and the other is coloured blue.

- Take all red vertices in the top layer as true literals and all blue vertices in the top layer as false literals.

Exercise. Check that the above true/false assignment to the variables in ϕ is a not-all-equal satisfying assignment.

5) Max-Cut $= \{(G,k) : G$ has a cut of size $\ge k\}$.

We will show that NAE-SAT $\le_p$ Max-Cut.

Let ϕ be the input 3CNF formula to NAE-SAT.
We will build a graph G_ϕ such that

$$\phi \in \text{NAE-SAT} \iff (G_\phi, 8m) \in \text{Max-Cut}$$

Here m is the number of clauses in ϕ.
Let n_i be the number of occurrences of $x_i / \bar{x}_i$ in ϕ.
So $\sum_i n_i = 3m$.

The graph G_ϕ has $2n$ vertices: $x_1, \ldots, x_n, \bar{x}_1, \ldots, \bar{x}_n$.
The edges are as follows:

* $2n_i$ parallel edges between x_i and $\bar{x}_i$ (for $1 \le i \le n$)
* a triangle for each clause in ϕ.

Claim 3. If $\phi \in$ NAE-SAT then $(G_\phi, 8m) \in$ Max-Cut.

Consider the cut $(S, V-S)$ where $S = \{$vertices that correspond to true literals$\}$ and $V-S = \{$vertices that corr. to false literals$\}$.

So for each i, we have x_i and $\bar{x}_i$ on opposite sides.
For each triangle: 2 vertices are on one side and 1 vertex on the opposite side.
So the number of edges crossing this cut $= \sum_i 2n_i + 2m = 8m$.

Claim 4. If $(G_\phi, 8m) \in$ Max-Cut then $\phi \in$ NAE-SAT.

We are given that G_ϕ has a cut with $8m$ edges crossing this cut. The only way we can have $8m$ edges crossing a cut is to have x_i and $\bar{x}_i$ on opposite sides: this will add up to $\sum_i 2n_i = 6m$ edges.

What about the remaining $2m$ edges? Observe that each triangle can contribute ≤ 2 edges to any cut. So to add up to $2m$ edges, each triangle has to contribute 2 edges to the cut – since there are m triangles, this adds up to $2m$.

View vertices on one side of the cut as true literals & vertices on the other side as false literals. Please check that this corresponds to an NAE-satisfying assignment.