NP-completeness

Thursday, April 14, 2022 1:35 PM

Categorize problems → easy problem
→ hard problems
→ decision problems → YES
→ NO

P = { problems that have polynomial time algorithms }
eg:- sorting, max-flow, matching, LPS

NP = { problems that have non-deterministic polynomial time algorithm }
→ if "YES" solution exist, then we can guess this solution in O(1) time.

eg:- SAT (satisfiability)

Given:- 1) variables $x_1, x_2, \ldots, x_n$
2) clauses, $C_1, C_2, \ldots C_m$

$$C_i = \{ x_{l_1} \vee \neg x_{l_2} \vee x_{l_3} \vee \ldots \}$$

Find:- ∃ assignment of {T, F} to
$\{x_1, x_2, \ldots x_n\}$ ...

$\{x_1, x_2, \ldots, x_n\}$ s.t. all clauses evaluate to T.

$F = C_1 \wedge C_2 \wedge \cdots \wedge C_m$

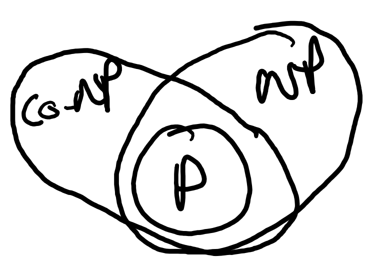
Non-deterministic alg for SAT

1) Guess
x_1 = T or F
x_2 = T or F
⋮
x_n = T or F
} $O(n)$

check whether all clauses are satisfied → YES / → NO } $O(nm)$

Relation between P & NP

$P \subseteq NP$



Big open problem $P = NP$?

... P = NP?

NP = {Problems that admit a polynomial size certificate/proof and a polynomial time verifier for all YES input}

Reduction: Given two decision problems A & B, a reduction is a mapping from all instances I of A to I′ of B s.t.

$$A(I) = \text{"YES"} \iff B(I') = \text{"YES"}$$

Polynomial-time-reduction (Karp-reduction)

Given 2 decision problems A & B, a poly-time-reduction is a poly.time algorithm that maps an instance I of A to I′ of B s.t.

$$A(I) = \text{"YES"} \iff B(I') = \text{"YES"}$$

$A \leq_p B$

Claim: $A \leq_p B$, then a poly-time alg for B implies a poly-time algorithm for A.

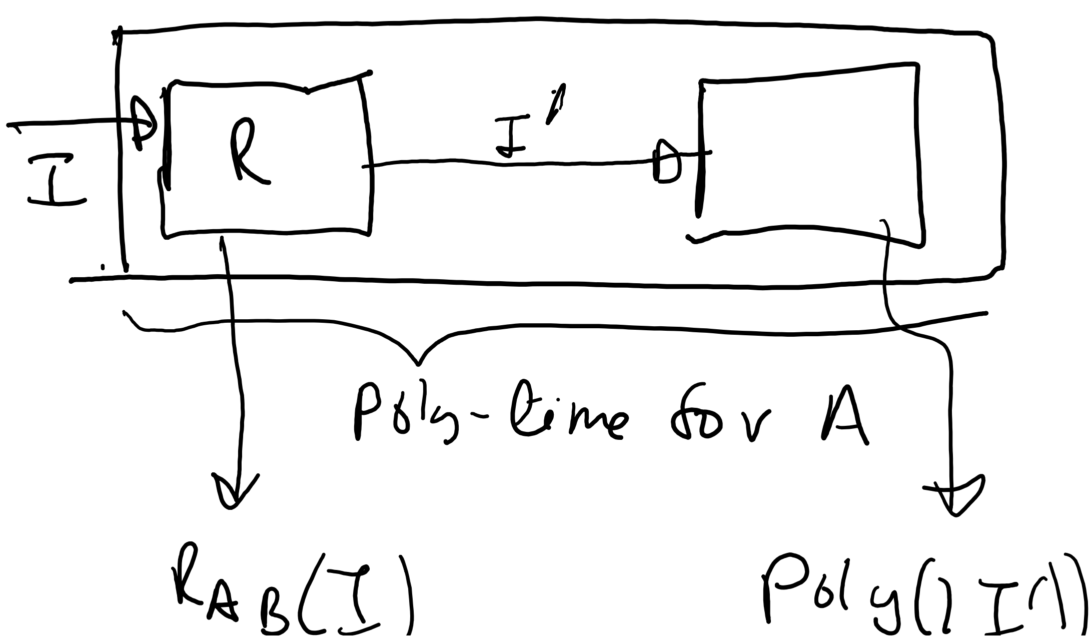
Pf:

OBS: Consider reduction R from A to B

R runs in time $R_{AB}(|I|) \in Poly(|I|)$

write at most $R_{AB}(|I|)$ bits

$\Rightarrow |I'| \leq R_{AB}(|I|) \in Poly(|I|)$



$R_{AB}(I)$ $\downarrow$ $\in Poly(|I|)$

$Poly(|I'|) \in Poly(poly(|I|)) \in poly(|I|)$

$\to Poly(I)$

Reductions are transitive:

$$A \leq_p B \ \& \ B \leq_p C$$
$$\Rightarrow A \leq_p C$$

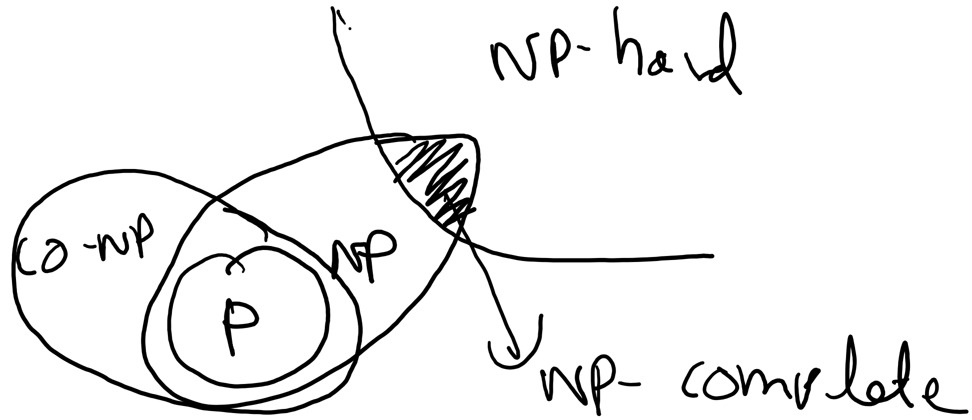
NP-hardness: $\Rightarrow$ a decision problem A is NP-hard if $\forall B \in NP$,

$$B \leq_p A$$

NP-completeness: $\Rightarrow$ a decision problem A is NP-complete iff

i) $A \in NP$

ii) A is NP-hard.



BIG-PICTURE

If A is NP-hard & $A \in P$

$\Rightarrow P = NP$

Problem C is NP-hard

$B \leq_p C \leq_p A$
$\in$
NP

Cook-Levin-Thm (1971 → Cook, 1973 → Levin)

SAT is NP-complete

3-SAT

Given: n variables $x_1, x_2, \ldots, x_n$

Given: variables $x_1, x_2, \ldots, x_n$

clauses $C_1, C_2, \ldots C_m$

$C_i = (x_{l_1} \vee x_{l_2} \vee \neg x_{l_3})$

Find: An assignment of {T, F} to all variables, s.t. all clauses are satisfied.

Thm: 3SAT is NP-complete

$SAT \leq_p 3SAT$.

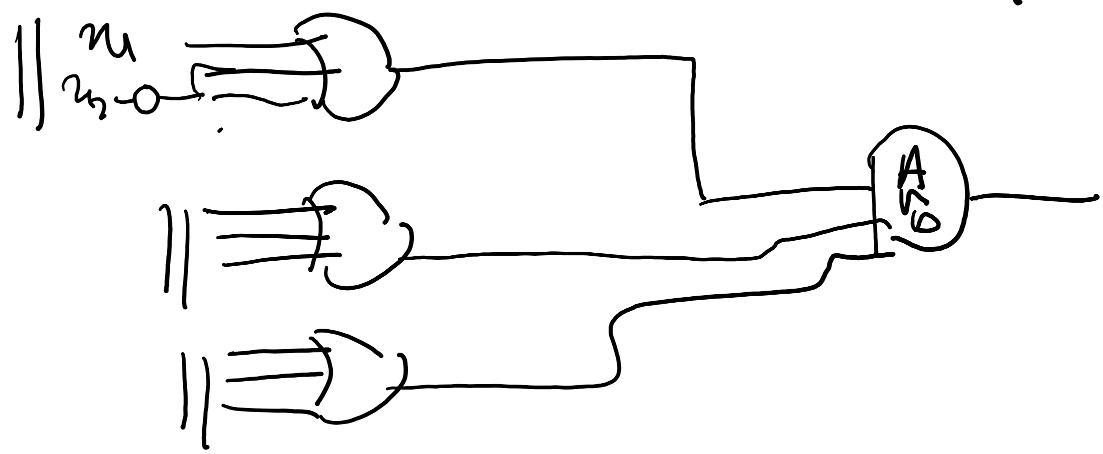
Goal: Find an algorithm (poly time) that converts an instance I of SAT to I' of 3SAT s.t.

I is satisfiable $\Leftrightarrow$ I' is satisfiable

$I = c_1 \wedge c_2 \wedge \cdots c_m$

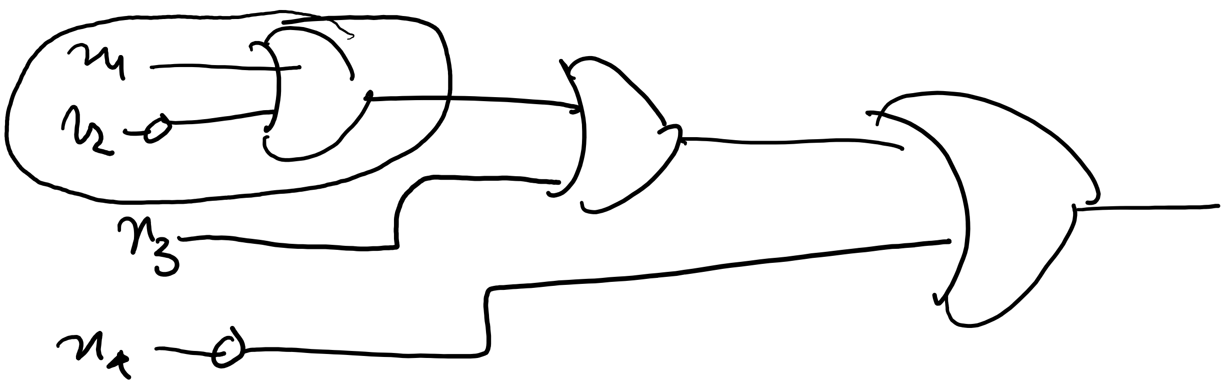
↳ $(x_1 \vee \neg x_2 \vee \cdots)$ upto n variables.

C

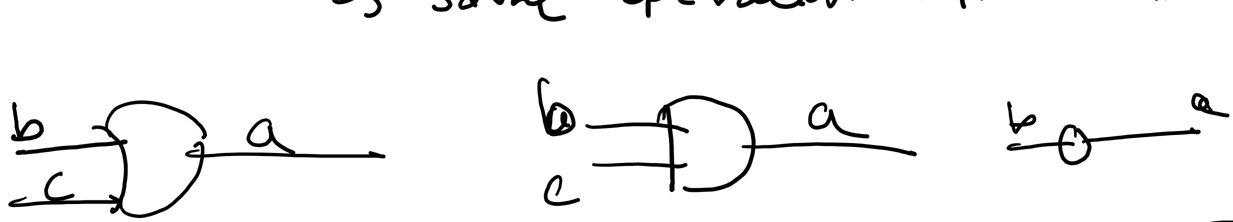


|

C′



↳ Same operation with AND



$[a = b \vee c] \wedge [a = b \wedge c] \wedge [a = \overline{b}]$

, OR GATE

1) $\overline{a = b \vee c} \iff (\bar{a} \vee b \vee c) \wedge (a \vee \bar{b}) \wedge (a \vee \bar{c})$

2) $a = b \wedge c \iff (a \vee \bar{b} \vee \bar{c}) \wedge (\bar{a} \vee b) \wedge (\bar{a} \vee c)$

3) $a = \bar{b} \iff (a \vee b) \wedge (\bar{a} \vee \bar{b})$

$I' = (\quad) \wedge (\quad) \wedge (\quad)$

upto 3 literals

clauses that have 3 literals = do nothing

clauses that have 2 literals

$(a \vee b) \Rightarrow (a \vee b \vee x) \wedge (a \vee b \vee \bar{x})$ replace

clauses that have 1 literal

$(a) \Rightarrow (a \vee x \vee y) \wedge (a \vee \bar{x} \vee y) \wedge$

$(a \vee \bar{x} \vee y)$
$(a \vee x \vee \bar{y})$
$(a \vee \bar{x} \vee \bar{y})$

I'

I is satisfiable ⟺ I' is satisfiable.

SAT $\leq_p$ 3SAT

3SAT is NP hard & SAT ∈ NP

⟹ 3SAT is NP complete

SAT $\leq_p$ 3SAT

Graph Theory (clique)

Given: Graph G with n vertices

& m edges & number k

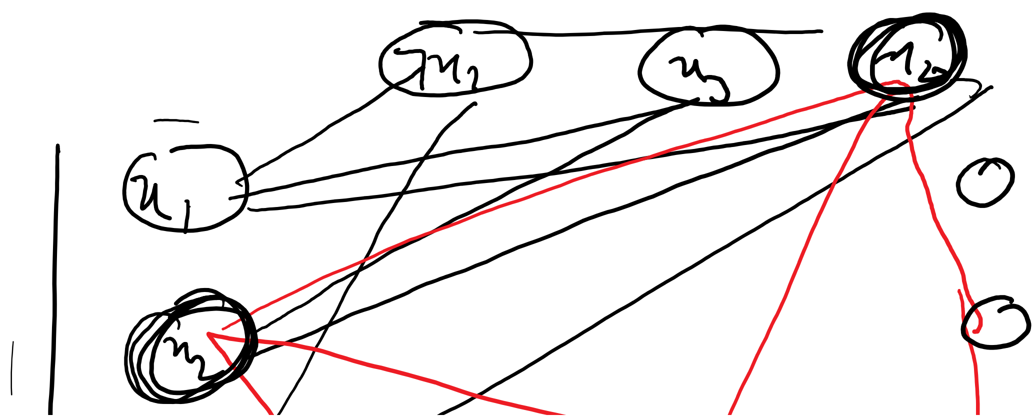
Question :- $\exists$ clique in G of size k ?

Thm :- clique is NP hard.

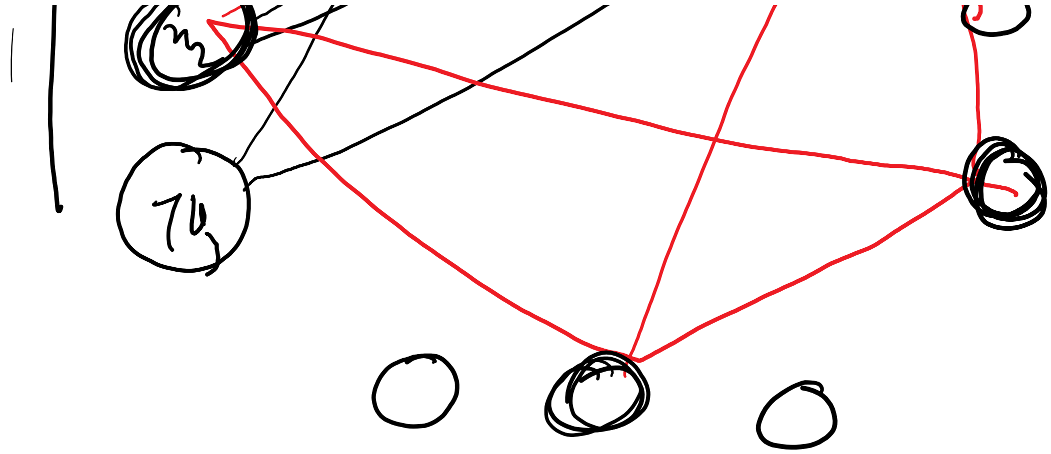
$SAT \leq_p 3SAT \leq_p CLIQUE$

Given an instance I of 3SAT
$\Rightarrow$ I' of CLIQUE in Polynomial time

$C_1 = (x_1 \vee x_2 \vee \neg x_3)$
$C_2 = (\neg x_2 \vee \neg x_3 \vee x_4)$
⋮
C_m



$G =$



Claim :- G has a clique of size m iff I has a satisfying assignment.

$\Rightarrow$ Look at a satisfying assignment. For each clause C, pick a literal that evaluates to T.

$\Leftarrow$ if $\exists$ a clique of size m, pick vertex x associated with clause C.

$u \rightarrow T$ if u is in C.

↳ F if $\bar{u}$ B in C

valid assignment

~~next lecture~~
2 lectures NP-completeness

↳ Graph problems
- → coloring
- → ind set
- ↳ vertex cover
- ↳ Hamiltonian

Optimization → knapsack
subset sum
ILP