NP-Completeness

Categorize problems

-> Easy problems

-hard problems

-> Decision problems

YES

NO

P= \{ problems that have polynomial time algorithm \}

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: 0) variables \(x_1, x_2, \ldots, x_n\)

2) clauses \(c_1, c_2, \ldots, c_m\)

\[ c_i = x_{i_1} \lor \ldots \lor x_{i_k} \]

Find: 0) Assignment of \( \{0, 1\} \) to \( x_1, x_2, \ldots, x_n \)

New Section 1 Page 1
\[ \{ u_1, u_2, \ldots, u_n \} : \text{all clauses evaluate to } T. \]

\[ F = C_1 \land C_2 \land \cdots \land C_m \]

Non-deterministic alg for SAT

1. Guess \[ u_1 \in T \text{ or } F \]
   \[ \begin{cases} 
   u_2 \in T \text{ or } F \\
   \vdots \\
   u_m \in T \text{ or } F
   \end{cases} \]

Check whether all clauses are satisfied

\[ \text{YES} \quad \text{if} \quad \text{all clauses are satisfied} \]

\[ \text{NO} \quad \text{otherwise} \]

Relation between \( P \) \& \( NP \)

\[ P \leq NP \]

Big open problem \( P = NP? \)
NP problems that admit a polynomial size certificate proof and a polynomial time verification for all yes inputs.

Reduction: Given two decision problem A and B, a reduction is a mapping from all yes cases I of A to I' of B such that:

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

Polynomial-time-reduction (Karp-reduction)

Given 2 decision problems A and B, a polynomial-time-reduction is a polynomial algorithm that maps an instance I of A to I' of B such that:

\[ A(I) = \text{"YES"} \iff B(I') = \text{"YES"} \]
Claim: $A \leq_p B$, then a poly-time algorithm for $B$ implies a poly-time algorithm for $A$.

Proof:

Observe: Consider reduction $R$ from $A$ to $B$.

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

Write at most $R_{AB}(I) \leq 6\cdot 5$

$= (|I|) \leq R_{AB}(I) \leq Poly(|I|)$

\[ R_{AB}(I) \leq Poly(|I|) \]
Reductions are transitive:

\[ A \leq_P B \land B \leq_P C \quad \Rightarrow \quad A \leq_P C \]

**NP-hardness:** A decision problem \( A \) is NP-hard if \( \forall B \in \text{NP}, B \leq_P A \)

**NP-completeness:** A decision problem \( A \) is NP-complete iff

1. \( A \in \text{NP} \)
2. \( A \) is NP-hard.
If $A$ is NP-hard and $A \in P$
\[ \implies P = NP \]

Problem $C$ is NP-hard

$B \leq_p C \leq_p A$


\[
\text{Cook–Levin Thm (1971)}
\]

SAT is NP-complete

3-SAT

Given $p$ variables $x_1, x_2, \ldots, x_n$
Given: \( p \) variables \( x_1, x_2, \ldots, x_n \)

clauses \( c_1, c_2, \ldots, c_m \)

\[ c_i = (x_1 \lor \neg x_2 \lor x_3) \]

Find: \( p \) an assignment of \( x_1, x_2, \ldots, x_n \)

all variables, s.t. all clauses are satisfied.

**Thm:** \( 3 \text{SAT} \) is \( NP \)-complete.

\[ \text{SAT} \leq_p 3 \text{SAT}. \]

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

\( I \) is satisfiable \( \iff \) \( I' \) is satisfiable.
I = \frac{c_i \land (a_2 \lor \ldots \lor a_m)}{[a_2 b v c] \lor [a_2 b v a] \lor [a_2 \overline{b}]} \quad \text{or GATE}
1) \[ a = bvc \iff (\bar{a}vb) \land (a \lor \bar{b}) \land (a \lor \bar{c}) \]

2) \[ a \neq bvc \iff (a \lor \bar{b} \lor \bar{c}) \land (\bar{a} \lor \bar{b}) \land (\bar{a} \lor v) \]

3) \[ a = b \iff (a \lor b) \land (\bar{a} \lor \bar{b}) \]

\[ T' = (\_ \lor \_ \lor \_) \land (\_ \lor \_ \lor \_ \lor \_ \lor \_ \lor \_ \lor \_ \lor \_) \]

\[ \text{upb is literally} \]

\[ \text{clauses that have 3 literals} \quad \text{do nothing} \]

\[ \text{clauses that have 2 literal} \quad (a \lor \bar{b}) \iff (a \lor b) \lor \bar{a} \lor \bar{b} \]

\[ \text{replace} \quad (a \lor b) \lor \bar{a} \lor \bar{b} \]

\[ \text{clauses that have 1 literal} \quad (a) \iff (a \lor \bar{u} \lor v \lor y) \lor (a \lor \bar{u} \lor \bar{v} \lor y) \lor (a \lor u \lor \bar{v} \lor y) \lor (a \lor u \lor v \lor \bar{y}) \lor (a \lor u \lor v \lor y) \lor (a \lor u \lor \bar{v} \lor y) \lor (a \lor u \lor v \lor \bar{y}) \lor (a \lor u \lor v \lor y) \]

\[ \vdash \]

New Section 1 Page 9
\( I \text{ is satisfiable} \implies I' \text{ is satisfiable.} \)

\[
\begin{align*}
\text{SAT} & \leq_p 3\text{SAT} \\
& \quad \quad \quad \text{(3SAT is } \text{NPHard} \quad \rightarrow \quad \text{3SAT is } \text{NPComplete}} \\
& \quad \quad \quad \quad \quad \quad \quad \quad \text{SAT} \in \text{NP} \\
& \quad \quad \quad \text{Graph Theory (clique)} \\
& \quad \quad \quad \quad \quad \quad \quad \quad \quad \text{Given: a Graph G with n vertices}
\end{align*}
\]
A new edge $e$ makes the number $k$ greater.

Question: Does there exist a clique in $G$ of size $k$?

Thm: A clique is $NP$-hard.

$SAT \leq_p 3SAT \leq_p CLIQUE$

Given an instance $I$ of $3SAT$

$= I'$ of $CLIQUE$ in $P$ (by construction)

$C_1 = (\overline{u_1} \lor u_2 \lor \overline{u_3})$

$C_2 = (\overline{u_1} \lor u_3 \lor \overline{u_2})$

$C_m$

Diagram of $C_1, C_2, \ldots, C_m$ with nodes $u_1, u_2, u_3, \ldots$ connected as per the clauses.
$G_i = \vdash$  

Claim: \[ G_i \text{ has a clique of size } m \iff I \text{ has a satisfying assignment.} \]

\[ \implies \] Look at a satisfying assignment. For each clause $c$, pick a literal that evaluates to $T$.

$\exists \text{ a clique of size } m \implies$ pick vertex $u$ associated with clause $c$. 

$u \not \in \{ \text{ a satisfying assignment} \}$
Valid assignment

Next lecture:
2. NP-completeness

Graph problem

Optimization

Linear Programming (LP)