\[ \text{NP-completeness} \]

Categorize problems \[ \rightarrow \text{non-tractable} \]

\[ \rightarrow \text{decision-problems} \]

\[ \text{ANSWER: "YES" "NO"} \]

\[ \text{max-flow} \rightarrow \exists f \text{ of capacity } F? \]

\[ \text{min-cut} \rightarrow \exists \text{ cut } S \text{ of size } k? \]

\[ P = \{ \text{set of problems solvable in polynomial time}\} \]

\[ \text{eg: max-flow, min-cut, shortest paths etc} \ldots \]

\[ \text{NP} = \{ \text{set of problems "verifiable" in polynomial time}\} \]

\[ \exists \text{ polynomial size proof/certificate } \]

\[ \text{a polynomial time algorithm to verify the solutions to "YES" instances.} \]

\[ \text{CNF-SAT.} \]
**CNF-SAT.**

Given:
1. A set of variables $x_1, x_2, \ldots, x_n$
2. A set of clauses $C_1, C_2, \ldots, C_m$.

$$C_i = \left( \overline{x_{i_1}} \lor x_{i_2} \lor \overline{x_{i_3}} \lor \ldots \right)$$

$C_i$ is satisfied if
- $x_{i_1} = T$, or $x_{i_2} = F$ or \ldots

**Question:** Does there exist an assignment to the variables such that all clauses are satisfied?

**Non-deterministic Alg**

1. **Guess** $x_1 = T$ or $F$
   - $x_2 = T$ or $F$
   - $x_n = T$ or $F$

**Verification Alg**

2. For each clause $C_i$, check whether $C_i$ is satisfied or not.
   - If satisfied, output YES
   - If not, output NO

$P \subseteq \text{NP}$

$p = \text{NP} \cap \text{co-NP}$
Reductions: Reduction $R$ is a mapping from instances of decision problem $A$ to instances of decision problem $B$ s.t.

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

Poly-time Reductions (Karp Reductions)

A poly-time ALG that maps any instance $I$ of $A$ to $I'$ of $B$, s.t.

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

$A \leq_{P} B$

Claim: If $I$ poly-time alg for $B$, then $I$ poly-time alg for $A$.

$R$: poly-time alg

$|I'| = \text{Poly}(|I|)$

Running time: $\text{Running time } (R) + \text{Running time } (ALG_{B})$
\[ A \leq_p B \]

**NP-hard**: Problem \( T \) is NP-hard if every problem \( A \in \text{NP} \), \( A \leq_p T \)

**NP-complete**: Problem \( A \) is NP-complete

1. \( A \in \text{NP} \)
2. \( A \) is NP-hard.

**Thm**: Cook-Levin Thm

**CNF-SAT is NP-complete**

\[
\text{CNF-SAT} \leq_p \text{1-CLIQUE} \leq_p \text{TSP}
\]

\[
\text{CNF-SAT} \leq_p \text{Hamiltonian} \leq_p \text{TSP}
\]
Given: 1) Undirected Graph \( G = (V,E) \)
2) Integer \( k \in V \)

Question: Does a clique of size \( k \) exist in \( G \).

**CNF-SAT \( \leq_p \) 3-SAT**

**Thm:** \( k \)-clique is \( \text{NP-complete} \)

1) clique \( \in \text{NP} \) (obvious)
2) 3-SAT \( \leq_p \) \( k \)-clique

We need a poly-time alg that maps every instance \( I \) of 3-SAT to \( I' \) of \( k \)-clique.

\[ C_1 = x_1 \vee \neg x_2 \vee x_3 \]
\[ C_2 = \neg x_2 \vee x_3 \vee x_4 \]

\( I \) is satisfiable \( \rightarrow \) \( I' \) has a clique of size \( k \).
⇒ I is satisfiable. 
For each clause \( C_r \), pick a literal \( l \in l_r \) that satisfies \( C_r \)

\[ G[l_1, \ldots, l_{km}, E] \models \text{clique} \]

∈ 
I' has a clique of size m
⇒ \( C \) has one vertex from each clause set.
⇒ Say \( l \) is chosen from \( C \) in
Set \( l \) to appropriate value to satisfy \( C_r \)

\[ IS: \text{GIVEN:}\quad \text{Und. Graph } G = (V,E) \]
\[ \text{integer } k \in [m] \]

\[ \text{Question:} \quad \exists IS \text{ of size } k \]
\[ S \subseteq V \text{ s.t } A VS \cup \nu \subseteq (V,V) \& E \]

\[ \text{Thm: } IS \text{ is NP-complete} \cdot \]
1) \( IS \in \text{NP} \text{ (obvious)} \)
2) \( k\text{-clique} \leq_\text{P} IS \)
I of $k$-clique $\Rightarrow$ $I'$ of IS

$$(G, E) \quad \leq \quad \overline{G} \quad \leq \quad k$$

**Hamiltonian Cycles:**

*Given:*
1) Directed Graph $G = (V, E)$
2) Exists cycle $C$ in $G$ that visits each part exactly once

**Thm:** $HC$ is NP-complete

1) $HC \in$ NP
2) 3SAT $\leq_p HC$

$(\eta_1, \eta_2, \ldots, \eta_n) =$ variable
$(c_1, c_2, \ldots, c_m) =$ clause

**Variable gadgets**

$$V = (\eta_1, \eta_2, \ldots, \eta_n)$$
Clause-gadgets

\[ V_{i, j, x} \rightarrow c_v \] if \( x \) occurs in \( c_v \).

\[ c_v \rightarrow V_{i, j, x+1} \] if \( x \) occurs in \( c_v \).

\[ c_v \rightarrow V_{j, x} \]

\[ V_{j, x+1} \rightarrow c_v \] if \( x \) occurs in \( c_v \).

\[ I \text{ has a satisfying assignment} \implies I' \text{ has a} \]

\[ HC \]

\[ \implies \text{straightforward.} \]

\[ \Leftarrow \text{let } c \text{ be a } HC \text{ in } I' \]

\[ \text{if } c \text{ enters } c_v \text{ then } V_{i, j, x} \text{ then it must leave } c_v \text{ to } V_{j, x+1} \]

\[ \text{Symmetry.} \]