---

---

## The P=NP Problem

Is P = NP?

Identify the *most difficult* problems in NP $\cdots$ NP-completeness

### SAT

A **Boolean formula** is a formula of propositional logic, constructed from variables and Boolean operations ($\wedge, \vee, \neg$)

A Boolean formula is **satisfiable** if there exists some assignment to the variables that makes the formula evaluate to 1

**Example:** $\phi = (\overline{x} \wedge y) \vee (x \wedge \overline{z})$ is satisfiable. A satisfying assignment is $x = 1, y = 1, z = 1$

$\phi = x \wedge \overline{x} \wedge y$ is not satisfiable.

---

## The Main Theorem

The **satisfiability problem** is the problem of deciding whether a given input Boolean formula is satisfiable

$SAT = \{\phi \mid \phi$ is a satisfiable Boolean formula $\}$

**Theorem.** *$SAT \in$ P if and only if P $=$ NP*

---

## Polynomial Time Reductions

**Definition.** *A function $f$ is polynomial time computable if there exists a polynomial time TM that halts on each input $x$ with only $f(x)$ on the tape.*

**Definition.** *A language $A$ is polynomial time mapping reducible to a language $B$ (write $A \leq_P B$) if $A$ is mapping reducible to $B$ via a polynomial time computable function.*

# 3SAT

A **literal** is a variable or its negation

A **clause** is the disjunction of some literals, e.g., $x_1 \vee \overline{x_3} \vee x_{17}$
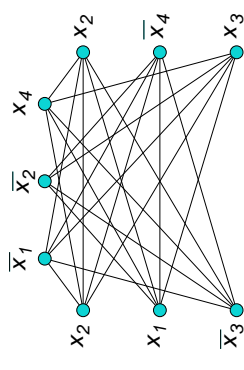
A Boolean formula is in **conjunctive normal form** if it is the conjunction ($\wedge$) of some clauses

A **3CNF formula** is a formula in CNF-form in which each clause consists of three literals

**Theorem.** *3SAT is polynomial time reducible to CLIQUE.*

---

# Proof of Theorem

Given a formula $\phi$ of $k$ three-literal clauses construct a graph $G = (V, E)$, where $V = \{\langle i,j \rangle \mid 1 \le i \le k, 1 \le j \le 3\}$ and $E = \{(\langle i,j \rangle, \langle i',j' \rangle) \mid (i \ne i')$ and **(the $j$th literal in the $i$th clause and the $j'$th literal in the $i'$th clause do not interfere with each other)**



The graph for the formula $\phi = (x_2 \vee x_1 \vee \overline{x_3}) \wedge (\overline{x_1} \vee \overline{x_2} \vee x_4) \wedge (x_2 \vee \overline{x_4} \vee x_3)$.

---

# Proof of Theorem (cont'd)

**Claim.** *G has a k-clique if and only if $\phi$ is satisfiable.*

[⇒] Let $S$ be a $k$-clique of $G$. Then $S$ **has a node from each triple** so that **the selected nodes do not interfere with each other** as assignments.

[⇐] Let $A$ be a satisfying assignment. Select from each triple **a literal that is satisfied by $A$** to construct a set $S$. $\| S \| = k$ and it is a clique.

The mapping is **polynomial time computable.**

---

# NP-Completeness

**Definition.** *A language $A$ is NP-complete if $A$ is in NP and every language in NP is polynomial time reducible to $A$.*

**Theorem.** *A language $A$ is NP-complete and $A \in P$, then $P = NP$.*

We may use the following to prove something is NP-complete.

**Theorem.** *A language $A$ is NP-complete, $B \in NP$, and $A \le_P B$, then B is NP-complete.*

# SAT is NP-Complete

**Theorem.** *SAT is* NP-*complete.*

**Proof** $SAT \in$ NP. Consider a two-tape NTM that, on an input $\phi$ of $n$ variables, **guesses and writes an assignment $A$** on Tape 2 using nondeterministic moves, accepts if $\phi(A) = 1$, and rejects $\phi(A) = 0$. Such a machine decides $SAT$ and can be polynomial time.

For the other part, suppose $A$ is in NP. We show $A \leq_P SAT$.

Since $A \in$ NP, there is some $k > 0$ and some $n^k$-time one-tape NTM $N$ such that $L(N) = A$. We will show that a deterministic TM can construct a formula within a polynomial time bound for any given input to $N$, describing accepting computations of $N$, where that formula is satisfiable iff there is such an accepting computation.

---

# Tableau

Let $w$ be an input of length $n$. Define a **tableau** for $N$ on $w$ to be an $n^k \times n^k$ table whose entries are from $\{\#\} \cup Q \cup \Gamma$ such that

1. **the columns $1$ and $n^k$ are all $\#$,**
2. **each row is a configuration of $N$,**
3. **the first row is the initial configuration of $N$ on $x$,** and
4. **for each $i$, $2 \leq i \leq n^k$, the $i$th row results from the $(i-1)$st row applying one move of $N$**

A tableau is **accepting** if

5. **the last row is an accepting configuration**

---

# Encoding of Tableau

Let $C = Q \cup \Gamma \cup \{\#\}$ and for each $i, j$, let $cell[i, j]$ denote the $j$th element in row $i$.

For each $i, j$, $1 \leq i, j \leq n^k$, and $s \in C$, let $x_{i,j,s}$ represent condition $cell[i, j] = s$.

Encode Conditions 1 through 5 into Boolean formulas $\phi_1$ through $\phi_5$ and let $x$ reduce to the conjunction of the five formulas.

**There is a one-to-one correspondence between the set of satisfying assignments and the set of all accepting tableaus**

---

**Condition 1**  "#'s"

$$\bigwedge_{1 \leq i \leq n^k} \left( x_{i,1,\#} \wedge x_{i,n^k,\#} \right).$$

**Condition 5**  "Accepting"

$$\bigvee_{1 \leq j \leq n^k} x_{n^k, j, q_{\text{accept}}}$$

**Condition 3**  "Initial"

$$x_{1,2,q_0} \wedge \left( \bigwedge_{1 \leq j \leq n} x_{1,2+j,w_j} \right) \wedge \left( \bigwedge_{n+3 \leq j \leq n^k} x_{1,2+j,\sqcup} \right)$$

**Condition 2** "configuration"

The conjunction of the following two for all $i$, $1 \leq i \leq n^k$:

$A_i$: $cell[i,j]$ has a unique value for all $j$:

$$\bigwedge_{1 \leq i,j \leq n^k} \left[ \left( \bigvee_{s \in C} x_{i,j,s} \right) \wedge \left( \bigwedge_{s,t \in C, s \neq t} (\overline{x_{i,j,s}} \vee \overline{x_{i,j,t}}) \right) \right]$$

$B_i$: There is precisely one $j$ such that $cell[i,j] \in Q$

$$\bigwedge_{1 \leq i \leq n^k} \left[ \left( \bigvee_{\substack{1 \leq j \leq n^k, \\ s \in Q}} x_{i,j,s} \right) \wedge \left( \bigwedge_{\substack{1 \leq j, \ell \leq n^k, \\ j \neq \ell}} \left( \left( \bigwedge_{s \in Q} \overline{x_{i,j,s}} \right) \vee \left( \bigwedge_{s \in Q} \overline{x_{i,\ell,s}} \right) \right) \right) \right]$$

---

**Condition 4** "step"

The conjunction of the following for all $i$, $2 \leq i \leq n^k$, $j$, $2 \leq j \leq n^k - 1$:

In the 2 block consisting of $cell[i-1, j-1], cell[i-1, j], cell[i-1, j+1], cell[i,j-1], cell[i,j], cell[i,j+1]$, if

($\alpha_{ij}$) $cell[i-1, j]$ is a state, then

($\beta_{ij}$) the values of the 6 cells correspond to one legal move of $N$;

and if

($\alpha'_{ij}$) $cell[i-1, j-1], cell[i-1, j], cell[i-1, j+1]$ are not states, then

($\beta'_{ij}$) $cell[i,j] = cell[i-1, j]$

Code this as $\overline{\alpha_{ij}} \vee \beta_{ij}$ and $\overline{\alpha'_{ij}} \vee \beta'_{ij}$.

The entire formula is of length bounded by some polynomial in $n$.

---

# 3SAT is NP-Complete

The formula in the tableau method can be **converted to an equivalent 3CNF formula**

**Conversion rules:**

1. $(x \wedge y) \wedge z$ is equivalent to $x \wedge y \wedge z$
2. $(x \vee y) \vee z$ is equivalent to $x \vee y \vee z$
3. $(x \vee y \vee z \vee u)$ is equivalent to $(x \vee y \vee w) \wedge (w \equiv (z \vee u))$. The second term is equivalent to $(\overline{w} \vee z \vee u) \wedge (\overline{z} \vee w) \wedge (\overline{u} \vee w)$

Repeat literals in clauses with $< 3$ literals to make the number of literals equal to three.