NP-Complete

CSCI 338
Can all problems that are verifiable in polynomial time be solved in polynomial time?

History Lesson:
- $P$ vs $NP$ concepts first discussed in 1950’s
- $P$ vs $NP$ formalized in 1971
SAT & 3SAT

\[ \phi = (x_1 \lor x_1 \lor x_2) \land (\overline{x_1} \lor \overline{x_2} \lor \overline{x_2}) \land (\overline{x_1} \lor x_2 \lor x_2) \]

\( \phi \) is a formula with clauses composed of Boolean variables connected by ORs, and clauses connected by ANDs.
$\phi = (x_1 \lor x_1 \lor x_2) \land (\overline{x_1} \lor \overline{x_2} \lor \overline{x_2}) \land (\overline{x_1} \lor x_2 \lor x_2)$

$\phi$ is a formula with clauses composed of Boolean variables connected by ORs, and clauses connected by ANDs.
\( SAT \ & \ 3SAT \)

\[
\phi = (x_1 \lor x_1 \lor x_2) \land (x_1 \lor x_2 \lor x_2) \land (x_1 \lor x_2 \lor x_2)
\]

\( \phi \) is a formula with clauses composed of Boolean variables connected by ORs, and clauses connected by ANDs.
SAT & 3SAT

\[ \phi = (x_1 \lor x_1 \lor x_2) \land (\overline{x_1} \lor \overline{x_2} \lor x_2) \land (\overline{x_1} \lor x_2 \lor \overline{x_2}) \]

\( \phi \) is a formula with clauses composed of Boolean variables connected by ORs, and clauses connected by ANDs.
SAT & 3SAT

\[ \phi = (x_1 \lor x_1 \lor x_2) \land (\overline{x_1} \lor \overline{x_2} \lor x_2) \land (\overline{x_1} \lor x_2 \lor \overline{x_2}) \]

\( \phi \) is a formula with clauses composed of Boolean variables connected by ORs, and clauses connected by ANDs.

(called conjunctive normal form – CNF)
**SAT & 3SAT**

\[ \phi = (x_1 \lor x_1 \lor x_2) \land (\overline{x_1} \lor \overline{x_2} \lor \overline{x_2}) \land (\overline{x_1} \lor x_2 \lor \overline{x_2}) \]

\( \phi \) is a formula with clauses composed of Boolean variables connected by ORs, and clauses connected by ANDs.

Can you set the variables to **true** or **false** so that \( \phi \) evaluates to **true**?
SAT & 3SAT

\[ \phi = (x_1 \lor x_1 \lor x_2) \land (\overline{x_1} \lor \overline{x_2} \lor \overline{x_2}) \land (\overline{x_1} \lor x_2 \lor x_2) \]

\[ x_1 = false \]
\[ x_2 = true \]
\[ \phi = (x_1 \lor x_1 \lor x_2) \land (\overline{x_1} \lor \overline{x_2} \lor \overline{x_2}) \land (\overline{x_1} \lor x_2 \lor x_2) \]

\[ (F \lor F \lor T) \quad (T \lor F \lor F) \quad (T \lor T \lor T) \]

\[ x_1 = false \]
\[ x_2 = true \]
SAT & 3SAT

\[ \phi = (x_1 \lor x_1 \lor x_2) \land (\overline{x_1} \lor \overline{x_2} \lor x_2) \land (x_1 \lor x_2 \lor x_2) \]

\[ (F \lor F \lor T) \quad (T \lor F \lor F) \quad (T \lor T \lor T) \]

\[ T \quad T \quad T \]

\[ x_1 = false \]

\[ x_2 = true \]
SAT & 3SAT

\[ \phi = (x_1 \lor x_1 \lor x_2) \land (\overline{x_1} \lor \overline{x_2} \lor \overline{x_2}) \land (\overline{x_1} \lor x_2 \lor x_2) \]

\[ (F \lor F \lor T) \quad (T \lor F \lor F) \quad (T \lor T \lor T) \]

\[ x_1 = false \]
\[ x_2 = true \]
\[ \phi = (x_1 \lor x_1 \lor x_2) \land \overline{(x_1 \lor x_2 \lor x_2)} \land (x_1 \lor x_2 \lor x_2) \]

\[
\begin{align*}
& (F \lor F \lor T) \\
& (T \lor F \lor F) \\
& (T \lor T \lor T)
\end{align*}
\]

- \(x_1 = false\)
- \(x_2 = true\)

\[ SAT = \{ ⟨\phi⟩: \phi \text{ is a satisfiable formula} \} \]
\[ 3SAT = \{ ⟨\phi⟩: \phi \text{ is a satisfiable formula with 3 variables per clause} \} \]
Cook-Levin Theorem: Every problem in $NP$ can be solved by a solver for $SAT$ with at most polynomial extra time.

Proof:
**Cook-Levin Theorem:** Every problem in $NP$ can be solved by a solver for $SAT$ with at most polynomial extra time.

**Proof:**

- **History Lesson:**
  - $P$ vs $NP$ concepts first discussed in 1950’s
  - $P$ vs $NP$ formalized in 1971
  - $SAT$ proven to solve everything in $NP$ in 1971
**P versus NP**

**Cook-Levin Theorem:** Every problem in \( NP \) can be solved by a solver for \( SAT \) with at most polynomial extra time.

**Proof:**

**History Lesson:**
- \( P \) vs \( NP \) concepts first discussed in 1950’s
- \( P \) vs \( NP \) formalized in 1971
- \( SAT \) proven to solve everything in \( NP \) in 1971
Cook-Levin Theorem: Every problem in $NP$ can be solved by a solver for $SAT$ with at most polynomial extra time.

Proof:

$P$ versus $NP$

**History Lesson:**
- $P$ vs $NP$ concepts first discussed in 1950’s
- $P$ vs $NP$ formalized in 1971
- $SAT$ proven to solve everything in $NP$ in 1971
NP-Complete

**History Lesson:**
- *P vs NP* concepts first discussed in 1950’s
- *P vs NP* formalized in 1971
- *SAT* proven to solve everything in *NP* in 1971
- *NP*-Complete defined in 1972

**SAT:**
The NP Super-Problem
**NP-Complete**

\[ B \text{ is in } NP-\text{Complete if it satisfies two conditions:} \]

1. \( B \in NP \).
2. For every \( A \in NP \), \( A \leq_p B \).

---

**History Lesson:**

- \( P \) vs \( NP \) concepts first discussed in the 1950’s.
- \( P \) vs \( NP \) formalized in 1971.
- \( SAT \) proven to solve everything in \( NP \) in 1971.
- \( NP \)-Complete defined in 1972.
$B$ is in $NP$-Complete if it satisfies two conditions:

1. $B \in NP$.
2. For every $A \in NP$, $A \leq_P B$.

"Every problem in $NP$ can be solved by an algorithm for $B$ in polynomial extra time."

**History Lesson:**
- $P$ vs $NP$ concepts first discussed in 1950’s
- $P$ vs $NP$ formalized in 1971
- SAT proven to solve everything in $NP$ in 1971
- $NP$-Complete defined in 1972
$B$ is in $NP$-Complete if it satisfies two conditions:
1. $B \in NP$.
2. For every $A \in NP$, $A \leq_P B$.

$B$ is in $NP$-Complete if it satisfies two conditions:
1. $B \in NP$.
2. For some $A \in NP-C$, $A \leq_P B$.

**History Lesson:**
- $P$ vs $NP$ concepts first discussed in 1950’s
- $P$ vs $NP$ formalized in 1971
- $SAT$ proven to solve everything in $NP$ in 1971
- $NP$-Complete defined in 1972
NP-Complete

$B$ is in $NP$-Complete if it satisfies two conditions:
1. $B \in NP$.
2. For every $A \in NP$, $A \leq_p B$.

History Lesson:
• $P$ vs $NP$ concepts first discussed in 1950's
• $P$ vs $NP$ formalized in 1971
• $SAT$ proven to solve everything in $NP$ in 1971
• $NP$-Complete defined in 1972
• 20 more problems shown to be in $NP$-Complete in 1972

SAT: The NP Super-Problem

$B$ is in $NP$-Complete if it satisfies two conditions:
1. $B \in NP$.
2. For some $A \in NP-C$, $A \leq_p B$. 
A solution to an *NP-Complete* problem can be used to solve any problem in *NP*, with just polynomial extra time.
A solution to an \textit{NP}-Complete problem can be used to solve any problem in \textit{NP}, with just polynomial extra time.
A solution to an $NP$-Complete problem can be used to solve any problem in $NP$, with just polynomial extra time.
A solution to an *NP*-Complete problem can be used to solve any problem in *NP*, with just polynomial extra time.

What if ∃ polynomial time algorithm for **Vertex Cover**?
A solution to an $NP$-Complete problem can be used to solve any problem in $NP$, with just polynomial extra time.

What if $\exists$ polynomial time algorithm for Vertex Cover?
- It could be used to solve any problem in $NP$ in polynomial time.
A solution to an \textit{NP}-Complete problem can be used to solve \textbf{any problem in NP}, with just polynomial extra time.

\textit{NP}-Complete Problems:
- Vertex Cover
- Independent Set
- SAT
- 3-SAT

What if \exists polynomial time algorithm for \textbf{Vertex Cover}?
- It could be used to solve \textbf{any problem in NP} in polynomial time.
- \( P = NP \).
**NP-Complete**

A solution to an *NP-Complete* problem can be used to solve any problem in *NP*, with just polynomial extra time.

What if $\exists$ polynomial time algorithm for *Vertex Cover*?
- It could be used to solve any problem in *NP* in polynomial time.
- $P = NP$. 

*NP-Complete Problems:*
- Vertex Cover
- Independent Set
- SAT
- 3-SAT
Are there problems in \(NP\), but not \(P\) or \(NP\)-Complete?

**P Problems:**
- Shortest Path
- Searching
- Sorting

**NP-Complete Problems:**
- Vertex Cover
- Independent Set
- SAT
- 3-SAT
Are there problems in $NP$, but not $P$ or $NP$-Complete?

- We don’t know. If so, $P \neq NP$.
- Suspected problems in $NP$ but not $P$ or $NP$-Complete:
  - Graph Isomorphism.
  - Integer Factorization.
$NP$-Complete
How to show something ($B$) is in $NP$-Complete?

\[ B \text{ is in } NP\text{-Complete if it satisfies two conditions:} \]
\begin{enumerate}
  \item $B \in NP$.
  \item For some $A \in NP\text{-C}$, $A \leq_p B$.
\end{enumerate}