NP Completeness:
Reductions

celebrating
40 years
of np-completeness
1971-2011

not “priced for quick sale”
"I can't find an efficient algorithm, I guess I'm just too dumb."
"I can’t find an efficient algorithm, because no such algorithm is possible!"
“I can’t find an efficient algorithm, but neither can all these famous people.”
Graph Coloring: assign colors to vertices so that adjacent vertices have different colors

4-colorable

not 3-colorable
f cannot be red, green or blue
Satisfiability: given a Boolean formula, does there exist an assignment of truth values to the variables that makes the formula true?

\[ P = (x_1 \lor \overline{x}_3 \lor \overline{x}_4) \land (\overline{x}_1 \lor x_2 \lor \overline{x}_4) \land (x_2 \lor \overline{x}_3 \lor x_4) \]

Satisfiable: \( x_1 = 1, x_2 = 0, x_3 = 0, x_4 = 0 \)

\[ P = (x_1 \lor x_2) \land (\overline{x}_1 \lor x_2) \land (x_1 \lor \overline{x}_2) \land (\overline{x}_1 \lor \overline{x}_2) \]

not satisfiable.

Note: truth table have an exponential # of rows.
Decision problems:

$3\text{COLOR} = \{ G \mid G$ is an undirected graph that is 3-colorable $\}$

$\text{SAT} = \{ \varphi \mid \varphi$ is a Boolean formula that is satisfiable $\}$

1. Given $G$, we can construct $\varphi'$ s.t.

   $G \in 3\text{COLOR} \iff \varphi' \in \text{SAT}$

   Easier if $\varphi$ is in 3CNF

2. Given $\varphi$, we can construct $G'$ s.t.

   $\varphi \in \text{SAT} \iff G' \in 3\text{COLOR}$

They look very different but are very much related.

If you have a fast algorithm for one, then you have a fast algorithm for the other.
G \rightarrow \phi' example

G has 9 vertices & 18 edges \rightarrow \phi' has 27 variables & 90 clauses

Colors: Red, Green, Blue

3 variables per vertex

\chi_R means vertex u is Red
\chi_G means vertex u is Green
\chi_B means vertex u is Blue

\begin{align*}
\{ & \chi_R, \chi_B, \chi_C, \chi_D, \chi_E, \chi_F, \chi_G, \chi_H, \chi_I \} \\
& \text{all 27 variables}
\end{align*}

Each vertex has at least 1 color

\phi' = \phi_1' \land \phi_2' \land \phi_3' \quad \text{adjacent vertices have different colors}

\text{no vertex has 2 colors}
\[ \varphi'_i = (X_a R \lor X_a G \lor X_a B) \land (X_b R \lor X_b G \lor X_b B) \land \ldots \land (X_i R \lor X_i G \lor X_i B) \]

Suppose that some assignment of 0-1 values to the variables makes \( \varphi'_i \) evaluate to 1.

Then in that assignment, every vertex has at least one color.

We want exactly 1
\[ \overline{q_2}' = (\overline{x_{AR}} \lor \overline{x_{AG}}) \land (\overline{x_{AR}} \lor \overline{x_{AB}}) \land (\overline{x_{AG}} \lor \overline{x_{AB}}) \]

- a not Red and Green at the same time
- a not Red and Blue at the same time
- a not Green and Blue at the same time

\[ (\overline{y} \lor \overline{z}): \text{if both } y = 1 \text{ and } z = 1, \text{ then } \overline{y} \lor \overline{z} = 0 \]

27 Clauses

\[ (\overline{x_{BR}} \lor \overline{x_{BG}}) \land \ldots \land (\overline{x_{iG}} \lor \overline{x_{iB}}) \]

If \( q_2' \) evaluates to 1, then assignment gives each vertex \(< 2 \) colors.

\( q_1' \land q_2' \) means each vertex has exactly 1 color.
We want to make sure that vertices a and vertex b have different colors.

- If a is Red $\implies$ b is not Red
  
  $(\overline{x_{ar}} \lor \overline{x_{br}})$

- a is Green $\implies$ b is not Green
  
  $\overline{x_{ag}} \lor \overline{x_{bg}}$

- a is Blue $\implies$ b is not Blue
  
  $\overline{x_{aB}} \lor \overline{x_{bB}}$

$$Q_3' = (\overline{x_{ar}} \lor \overline{x_{br}}) \land (\overline{x_{ag}} \lor \overline{x_{bg}}) \land (\overline{x_{aB}} \lor \overline{x_{bB}}) + a \land b$$

Different colors in a satisfying assign.

$18 \times 3$

$= 54$ clauses

$$\land (\overline{x_{hR}} \lor \overline{x_{iR}}) \land (\overline{x_{hG}} \lor \overline{x_{iG}}) \land (\overline{x_{hB}} \lor \overline{x_{iB}})$$

Different colors h & i
Recap: $\varphi' = \varphi_1 \land \varphi_2 \land \varphi_3$  
27 variables  
$(9 + 27 + 54) = 90$ clauses  

Need to argue $G \in 3\text{COLOR} \iff \varphi' \in \text{SAT}$  

$(\Rightarrow)$ If $G$ is 3-colorable, then every vertex of $G$ is assigned one of $\{\text{Red, Green, Blue}\}$  
For each vertex $u$, if $u$ is assigned Red, $x_u = 1$, $x_u = 0$, $x_u = 0$  
if $u$ is assigned Green, $x_u = 0$, $x_u = 1$, $x_u = 0$  
if $u$ is assigned Blue, $x_u = 0$, $x_u = 0$, $x_u = 1$.  
$\varphi_1$ and $\varphi_2$ are satisfied since each vertex has exactly one color.  
$\varphi_3$ is satisfied because we started w/ a legal coloring of $G$.  
No two adjacent vertices are assigned the same color.
(\Leftarrow)

For all vertices $u$, exactly one of $x_{ur}, x_{ua} \& x_{ub}$ is assigned 1, since $\phi_1'$ and $\phi_2'$ are both satisfied. Assign vertex $u$ the corresponding color. Since $\phi_3'$ is also satisfied, no two adjacent vertices are assigned same color.

$G$ is 3-colorable $\Rightarrow \phi'$ is satisfiable

$G$ is not 3-colorable $\Rightarrow \phi'$ is not satisfiable.

If we have an algorithm for SAT, then we can use it to determine if a graph is 3-colorable.
Recap: Efficient transformation of an instance of 3-COLOR into an instance of SAT.

**Defn:** we say that 3COLOR reduces to SAT.

**Idea:** if there's an efficient algorithm $A$ for SAT, then there's an efficient algorithm for 3COLOR

1. $G \mapsto \phi$
2. Run $A$ on $\phi$
3. If $A$ says $\phi \in \text{SAT}$, output YES
4. o.w. Output NO

What if SAT is much harder than 3-COLOR?
Given a Boolean formula $\Phi$ in 3CNF, construct an undirected graph $G'$ s.t.

$\Phi \in \text{SAT} \iff G' \in \text{3COLOR}$

Some "gadgets":

- palette
- equals
- negate

red = true  
blue = false
Clause Gadget:

- blue = false
- red = true

- $x_1, x_2, x_3$ must be blue or red in any 3-coloring

- If $x_1, x_2, x_3$ are all blue, output must be blue.

- If one of $x_1, x_2, x_3$ is red, then output can be red.
transform $\varphi = (x_1 \lor \overline{x}_3 \lor \overline{x}_4) \land (\overline{x}_1 \lor x_2 \lor \overline{x}_4) \land (\overline{x}_2 \lor \overline{x}_3 \lor x_4)$ into a 3COLOR problem.
Recap: We can efficiently transform an instance \( \phi \) of SAT into an instance \( G' \) of 3COLOR s.t.

\[ \phi \in \text{SAT} \iff G' \in \text{3COLOR} \]

I.e., we reduced SAT to 3COLOR.

Previously, we reduced 3COLOR to SAT.

\( \therefore \) 3COLOR and SAT have equivalent complexity.

They are both NP-complete.
"I can’t find an efficient algorithm, but neither can all these famous people."