REducing one decision problem to another
REDUCING ONE (DECISION) PROBLEM TO ANOTHER

input for A ➔ Transform to input/problem B ➔ solve B ➔ YES or NO

output of B ➔ YES or NO

output of A ➔ YES or NO

solve A using B
REDUCING ONE (DECISION) PROBLEM TO ANOTHER

input for A → Transform to input/problem B → solve B → output of B → YES/NO

solve A using B : T_R + T_B

output of B → YES/NO

output of A → YES/NO

input for A
REDUCE ONE (DECISION) PROBLEM TO ANOTHER

input for A

Transform to input/problem B

solve B

output of B

no

yes

solve A using B : \( T_R + T_B \)

is L.I.S. (A) \( \geq k \) ?

no

yes
REDDUCING ONE (DECISION) PROBLEM TO ANOTHER

INPUT FOR A

Transform to input/problem B

Solve B

Output of B

Solve A using B : TR + TB

Is L.I.S. (A) ≥ k ?

Copy A → B, sort B

Is L.C.S. (A, B) ≥ k ?

OUTPUT OF A

YES

NO

YES

NO

YES

NO

YES

NO
LOWER BOUND FOR ONE (DECISION) PROBLEM VIA ANOTHER

input for D

solve D: known $\Omega(f(n))$

output of D

C

D

YES

NO
LOWER BOUND FOR ONE (DECISION) PROBLEM VIA ANOTHER

input for D

Transform to input/problem C

solve C

output of C

output of D

solve D: known $\Omega(f(n))$
LOWER BOUND FOR ONE (DECISION) PROBLEM VIA ANOTHER

input for D

Transform to input/problem C

$T_R = o(f(n))$

solve C

output of C

$T_C$

output of D

solve D: known $\Omega(f(n)) = T_R + T_C$
Lower bound for one (decision) problem via another

\[ T_R = o(f(n)) \]

\[ T_C = \Omega(f(n)) \]

\[ \text{solve } D: \text{ known } \Omega(f(n)) = T_R + T_C \Rightarrow T_C = \Omega(f(n)) \]
LOWER BOUND FOR ONE (DECISION) PROBLEM VIA ANOTHER

\[ T_R = o(f(n)) \]

[Diagram]

Transform to input/problem \( C \)

Solve \( C \)

output of \( C \)

Yes \( \Rightarrow \) output of \( D \)

Yes \( \Rightarrow \) No

No \( \Rightarrow \) Yes

solve \( D \): known \( \Omega(f(n)) = T_R + T_C \Rightarrow T_C = \Omega(f(n)) \)

- quick transformation \( \Rightarrow \) \( C \) is at least as difficult as \( D \)
  (fast solution for \( C \) \( \Rightarrow \) fast solution for \( D \))
NPC Reduction for one (decision) problem via another
NPC REDUCTION FOR ONE (DECISION) PROBLEM VIA ANOTHER

input for D → D: known NPC

output of D → YES

output of D → NO
NPC REDUCTION FOR ONE (DECISION) PROBLEM VIA ANOTHER

input for D

Transform to input/problem C

solve C

output of C

YES

output of D

YES

NO

NO

D: known NPC
NPC REDUCTION FOR ONE (DECISION) PROBLEM VIA ANOTHER

D: known NPC

input for D

Transform to input/problem C

TR = polynomial

solve C

output of C

D

output of D

YES

NO

YES

NO
NPC REDUCTION FOR ONE (DECISION) PROBLEM VIA ANOTHER

Input for D

Transform to input/problem C

$T_R = \text{polynomial}$

Solve C

Solve D: known NPC = $T_R + T_C = \text{poly-time} + T_C$

Output of C

Yes

No

Output of D

Yes

No
NPC REDUCTION for one (decision) PROBLEM via ANOTHER

input for D

transform to input/problem C

$T_R = \text{polynomial}$

solve C

$T_C$

output of C

output of D

yes

no

\(\Rightarrow T_C \text{ is NP-hard} \)  
(or $T_C$ is polynomial $\Rightarrow P = NP$)
**NPC Reduction for One (Decision) Problem via Another**

**Input for D**

Transform to input/problem C

$T_R = \text{polynomial}$

Solve C

Output of C

Solve D: known NPC

$T_R + T_c = \text{poly-time} + T_c$

$\Rightarrow T_c$ is NP-hard

& if C is in NP

then C is NPC

(or $T_c$ is polynomial $\Rightarrow P=NP$)
NPC REDUCTION for one (decision) problem via another

~like establishing an almost certain lower bound for C.

\[ T_{R} = \text{polynomial} \]

solve D: known NPC

\[ T_{R} + T_{C} = \text{poly-time} + T_{C} \]

⇒ \( T_{C} \) is NP-hard

(\( \text{or } T_{C} \text{ is polynomial} \Rightarrow P=NP \))
$A \leq_p B$ : in polynomial time, $A$ can be transformed to $B$
$A \leq_p B$ : in polynomial time, $A$ can be transformed to $B$

$\Rightarrow B$ is at least as hard as $A$.

Solving $B$ in poly-time $\Rightarrow$ solving $A$ in poly-time
$A \leq_p B$ : in polynomial time, $A$ can be transformed to $B$

$\implies B$ is at least as hard as $A$.
Solving $B$ in poly-time $\implies$ solving $A$ in poly-time

If $\{\text{any problem in NP}\} \leq_p B$ then $B$ is NPC.
\[ A \leq_p B : \text{ in polynomial time, } A \text{ can be transformed to } B \]

\[ \Rightarrow B \text{ is at least as hard as } A. \]

Solving \( B \) in poly-time \( \Rightarrow \) solving \( A \) in poly-time

If \( \{\text{any problem in } \text{NP}\} \leq_p B \), then \( B \) is NPC.

Notice if \( A \) & \( B \) are NPC then \( A \leq_p B \) & \( B \leq_p A \).

\( \sim A \equiv_p B \)
Thousands of NPC problems
Thousands of NPC Problems

Reductions/transformations between them resemble a di-graph.
THOUSANDS OF NPC PROBLEMS

Reductions/transformations between them resemble a di-graph.

\( \not \in \text{a DAG? No.} \)
Thousands of NPC Problems

Reductions/transformations between them resemble a di-graph.

A DAG? No.

# strongly connected components?
THOUSANDS OF NPC PROBLEMS

Reductions/transformations between them resemble a di-graph.

Is a DAG? No.

# Strongly connected components?
Must be 1, precisely.
THOUSANDS OF NPC PROBLEMS

Reductions/transformations between them resemble a di-graph.

≠ a DAG? No.

#strongly connected components?

Must be 1, precisely.

FACT

∃ at least one problem A s.t.
A ≽p everything?
& everything ≽p...≽p A
Circuit SAT (satisfiability)

The first NPC problem.
Circuit SAT (satisfiability)
The first NPC problem.
Given a circuit, can the output ever be 1?
Circuit SAT (satisfiability)
The first NPC problem.
Given a circuit, can the output ever be 1?

\[ \begin{cases} 
0 & x_1 \\
1 & x_2 \\
1 & x_4 \\
0 & x_5 \\
\end{cases} \]

\begin{align*}
\text{OR} & \quad \text{OR} \\
\text{AND} & \\
\text{AND} & \\
\end{align*}

\text{output}
Circuit SAT (satisfiability)
The first NPC problem.
Given a circuit, can the output ever be 1?

\[ x_i : \text{input} \]

\[ \begin{align*}
0 & \quad x_1 \quad \text{OR} \quad 1 \\
1 & \quad x_2
\end{align*} \]

\[ \begin{align*}
0 & \quad x_3 \quad \text{NOT} \quad 1 \\
1 & \quad x_4 \quad \text{AND} \\
0 & \quad x_5
\end{align*} \]

\[ \text{AND} \quad \text{output} \]
Circuit SAT (satisfiability)
The first NPC problem.
Given a circuit, can the output ever be 1?

\[ x_1, x_2, x_3, x_4, x_5 \]

...
Circuit SAT (satisfiability)
The first NPC problem.
Given a circuit, can the output ever be 1?
Circuit SAT (satisfiability)
The first NPC problem.
Given a circuit, can the output ever be 1?

\[ x_i \text{: input} \]

\[
\begin{align*}
0 & \quad x_1 \quad \text{OR} \\
0 & \quad x_2 \\
0 & \quad \text{OR} \\
0 & \quad x_3 \quad \text{NOT} \\
0 & \quad x_4 \\
0 & \quad x_5 \\
\end{align*}
\]

\[
\begin{align*}
\text{AND} & \\
\text{AND} & \quad \text{(yes)} \\
\text{output} & \quad 1
\end{align*}
\]
Circuit SAT (satisfiability)
The first NPC problem.
Given a circuit, can the output ever be 1?
1 - in NP ... intuitive
Circuit SAT (satisfiability)
The first NPC problem.
Given a circuit, can the output ever be 1?

1 - in NP ... intuitive
2 - every problem can be described as a circuit
\{everything\} \leq \text{p} \text{ circuit-SAT}
Circuit SAT (satisfiability)
The first NPC problem.
Given a circuit, can the output ever be 1?

1 - in NP ... intuitive
2 $\forall$ every problem can be described as a circuit \{technical
$\forall$everythings $\leq_p$ circuit-SAT

...and we don't know how to solve this in polynomial time. (not in P)
Boolean SAT \quad (x_1 \lor x_2 \lor \overline{x}_3) \land ((x_1 \leftrightarrow x_5) \lor (x_4 \rightarrow \overline{x}_3)) \rightarrow 1
Boolean SAT: \((x_1 \lor x_2 \lor \overline{x}_3) \land ((x_1 \leftrightarrow x_5) \lor (x_4 \rightarrow \overline{x}_3)) \rightarrow \neg \rightarrow 1\)

Given a circuit-SAT instance:

[Diagram of a circuit with nodes and gates]
Boolean SAT \((x_1 \lor x_2 \lor x_3) \land (x_1 \leftrightarrow x_5) \lor (x_4 \rightarrow \overline{x}_3)\) \(\rightarrow \) ? \(\rightarrow 1\)

Given a circuit-SAT instance, transform (quickly) into a Boolean SAT.

\[ w \land (w \leftrightarrow (y_4 \land y_2 \land y_3)) \]
Boolean SAT \((x_1 \lor x_2 \lor \overline{x_3}) \land ((x_1 \leftrightarrow x_5) \lor (x_4 \rightarrow \overline{x_3}))\) → ? → 1

Given a circuit-SAT instance, transform (quickly) into a Boolean SAT:

\[ w \land (w \leftrightarrow (y_4 \land y_2 \land y_3)) \land (y_4 \leftrightarrow (y_1 \lor y_2)) \]
Given a circuit-SAT instance, transform (quickly) into a Boolean SAT.

\[
\begin{align*}
    \text{Boolean SAT} & \quad (x_1 \lor x_2 \lor \overline{x_3}) \land ((x_1 \leftrightarrow x_5) \lor (x_4 \rightarrow \overline{x_3})) \\
    \text{transform} & \quad \rightarrow \quad 1
\end{align*}
\]
Given a circuit-SAT instance, transform (quickly) into a Boolean SAT:

\[
\begin{align*}
  &\neg \phi \\
  &= \neg (w \land (w \leftrightarrow (y_4 \land y_2 \land y_3))) \\
  &\quad \land (y_4 \leftrightarrow (y_1 \lor y_2)) \\
  &\quad \land (y_1 \leftrightarrow (c_1 \lor c_2)) \\
  &\quad \land (y_2 \leftrightarrow \overline{c_3})
\end{align*}
\]
Boolean SAT: 

\[(x_1 \lor x_2 \lor \overline{x}_3) \land ((x_1 \leftrightarrow x_5) \lor (x_4 \rightarrow \overline{x}_3)) \rightarrow ? \rightarrow 1\]

Given a circuit-SAT instance, transform (quickly) into a Boolean SAT.

\[w \land (w \leftrightarrow (y_4 \land y_2 \land y_3))\]
\[\land (y_4 \leftrightarrow (y_1 \lor y_2))\]
\[\land (y_1 \leftrightarrow (c_1 \lor c_2))\]
\[\land (y_2 \leftrightarrow \overline{c_3})\]
\[\land (y_3 \leftrightarrow (c_4 \land c_5))\]
Boolean SAT: 

\((x_1 \lor x_2 \lor \overline{x_3}) \land ((x_1 \leftrightarrow x_5) \lor (x_4 \rightarrow \overline{x_3})) \rightarrow ? \rightarrow 1\)

Given a circuit-SAT instance transform (quickly) into a Boolean SAT.

by solving the Boolean SAT you get an answer for circuit SAT:

\(w \land (w \leftrightarrow (Y_4 \land Y_2 \land Y_3))\)

\(\land (Y_4 \leftrightarrow (Y_1 \lor Y_2))\)

\(\land (Y_1 \leftrightarrow (C_1 \lor C_2))\)

\(\land (Y_2 \leftrightarrow \overline{C_3})\)

\(\land (Y_3 \leftrightarrow (C_4 \land C_5))\)
3-SAT

\[ (x_1 \lor x_2 \lor x_3) \]
\[ \land (x_1 \lor x_4 \lor x_5) \]
\[ \land (x_2 \lor \overline{x}_4 \lor x_{13}) \]
\[ \land (x_2 \lor \overline{x}_3 \lor x_3) \]
\[ \vdots \]

\( n \) clauses
\( \) each with 3 literals
\[(a \lor b \lor c) \land (b \lor \overline{c} \lor \overline{d}) \land (\overline{a} \lor c \lor d) \land (a \lor \overline{b} \lor \overline{d})\] 

\(k\) clauses

3-SAT \rightarrow boolean SAT \rightarrow Circuit SAT

construct graph (quickly)
\((a \lor b \lor c) \land (b \lor \overline{c} \lor \overline{d}) \land (\overline{a} \lor c \lor d) \land (a \lor \overline{b} \lor \overline{d})\) 

\(k\) clauses

3-SAT \rightarrow boolean SAT \rightarrow circuit SAT

construct graph (quickly)

for every triple \((x \lor y \lor z)\) make a triple of vertices, linked together
\[(a \lor b \lor c) \land (b \lor \overline{c} \lor \overline{d}) \land (\overline{a} \lor c \lor d) \land (a \lor \overline{b} \lor \overline{d})\] 

\(k\) clauses

Construct graph (quickly)
\[(a \lor b \lor c) \land (b \lor \overline{c} \lor \overline{d}) \land (\overline{a} \lor c \lor d) \land (a \lor \overline{b} \lor \overline{d})\]\n
\(3\text{-SAT} \rightarrow \text{boolean SAT} \rightarrow \text{circuit SAT}\)

\(k\) clauses

construct graph (quickly)
\[(a \lor b \lor c) \land (b \lor \overline{c} \lor \overline{d}) \land (\overline{a} \lor c \lor d) \land (a \lor \overline{b} \lor \overline{d})\]

\[\text{construct graph (quickly)}\]

\[\text{link every vertex to any other vertex that has the same variable but negated}\]
\[(a \lor b \lor c) \land (b \lor \overline{c} \lor \overline{d}) \land (\overline{a} \lor c \lor d) \land (a \lor \overline{b} \lor \overline{d})\] k clauses

3-SAT \rightarrow boolean SAT \rightarrow circuit SAT

Ask: does this graph have k independent vertices?

To find k independent, you must use each triple, because each triple can only have 1 independent vertex.
As soon as you select a vertex in a triple, that blocks you from using its negation anywhere else.
\[(a \lor b \lor c) \land (b \lor \bar{c} \lor \bar{d}) \land (\bar{a} \lor c \lor d) \land (a \lor \bar{b} \lor \bar{d})\] 

\(k\) clauses

If you can't find \(k\) independent vertices, then some triple had no selectable vertex... i.e. all 3 were blocked by selections in other triples. That corresponds to an unsatisfiable row in 3-SAT.

Ask: does this graph have \(k\) independent vertices?

If yes, then 3-SAT is \(\checkmark\)
If no, then 3-SAT is \(\times\)
\[(a \lor b \lor c) \land (b \lor \bar{c} \lor \bar{d}) \land (\bar{a} \lor c \lor d) \land (a \lor \bar{b} \lor \bar{d})\] 

\(k\) clauses

Construct graph (quickly)

Ask: does this graph have \(k\) independent vertices?

If yes, then 3-SAT is \(\checkmark\)

If no, then 3-SAT is \(\times\)
\[(a \lor b \lor c) \land (b \lor \overline{c} \lor \overline{d}) \land (\overline{a} \lor c \lor d) \land (a \lor \overline{b} \lor \overline{d})\] 

\[k \text{ clauses}\]

\[3\text{-SAT} \rightarrow \text{boolean SAT} \rightarrow \text{Circuit SAT}\]

\[\text{independent set}\]

\[\text{Ask: does this graph have } k \text{ independent vertices?}\]

\[\text{If yes, then 3-SAT is } \checkmark\]
\[\text{If no, then 3-SAT is } \times\]
CLIQUE in a graph: subset of $V$ s.t. all pairs of vertices share edges.
CLIQUE in a graph: subset of $V$ s.t. all pairs of vertices share edges.
CLIQUE in a graph: subset of $V$ s.t. all pairs of vertices share edges.

Independent set size $k$ $\iff$ Cliquer size $k$
CLIQUE in a graph: subset of $V$ s.t. all pairs of vertices share edges.

independent set size $k$ $\iff$ clique size $k$

$G$ $\xrightarrow{\text{transform}}$ complement $(G)$

Asking if a graph has a clique of size $k$: NPC
OTHER HARD PROBLEMS

- knapsack: given item types, w/ size & value, fill a bag w/ max value (multiples ok)
- subset sum: does a subset of given integers sum to t?
- partition: split set of integers in 2 groups with equal sums
- planar SAT: SAT without wire crossings
- set cover: given some sets, select min# sets s.t. all elements are present
- hitting set: given sets, select min# elements s.t. all sets are represented
- longest path: visiting each vertex once.
- Steiner tree: ~MST on selected subset of a graph
- traveling salesman: min-cost simple cycle on weighted complete graph
OTHER HARD PROBLEMS

& some are even harder

Tetris Minesweeper Lemmings

Mario Bros Pac-man

Prince of Persia Portal Doom

etc