Classify problems according to computational requirements

Q. Which problems will we be able to solve in practice?

A working definition. Those with polynomial-time algorithms.

Theory. Definition is broad and robust.

Practice. Poly-time algorithms scale to huge problems.
Polynomial-time reductions

Desiderata. Suppose we could solve problem $Y$ in polynomial-time. What else could we solve in polynomial time?

Reduction. Problem $X$ polynomial-time (Cook) reduces to problem $Y$ if arbitrary instances of problem $X$ can be solved using:

- Polynomial number of standard computational steps, plus
- Polynomial number of calls to oracle that solves problem $Y$.

![Diagram of reduction process]

computational model supplemented by special piece of hardware that solves instances of $Y$ in a single step
Polynomial-time reductions

Desiderata. Suppose we could solve problem $Y$ in polynomial-time. What else could we solve in polynomial time?

Reduction. Problem $X$ polynomial-time (Cook) reduces to problem $Y$ if arbitrary instances of problem $X$ can be solved using:

- Polynomial number of standard computational steps, plus
- Polynomial number of calls to oracle that solves problem $Y$.

Notation. $X \leq_p Y$.

Note. We pay for time to write down instances sent to oracle $\Rightarrow$ instances of $Y$ must be of polynomial size.

Caveat. Don't mistake $X \leq_p Y$ with $Y \leq_p X$.

$X, Y$ NP-complete: $X \leq_p Y$ 

$X \leq_p Y$ 

$X \leq_p Y$ 

$X \leq_p Y$
Polynomial-time reductions

**Design algorithms.** If $X \leq_p Y$ and $Y$ can be solved in polynomial time, then $X$ can be solved in polynomial time.

**Establish intractability.** If $X \leq_p Y$ and $X$ cannot be solved in polynomial time, then $Y$ cannot be solved in polynomial time.

**Establish equivalence.** If both $X \leq_p Y$ and $Y \leq_p X$, we use notation $X =_p Y$. In this case, $X$ can be solved in polynomial time iff $Y$ can be.

**Bottom line.** Reductions classify problems according to relative difficulty.
Independent set

**INDEPENDENT-SET.** Given a graph $G = (V, E)$ and an integer $k$, is there a subset of vertices $S \subseteq V$ such that $|S| \geq k$, and for each edge at most one of its endpoints is in $S$?

**Ex.** Is there an independent set of size $\geq 6$?
**Ex.** Is there an independent set of size $\geq 7$?

![Graph with independent set of size 6]

\[ \text{independent set of size 6} \]
**Vertex cover**

**vertex-cover.** Given a graph $G = (V, E)$ and an integer $k$, is there a subset of vertices $S \subseteq V$ such that $|S| \leq k$, and for each edge, at least one of its endpoints is in $S$?

**Ex.** Is there a vertex cover of size $\leq 4$?

**Ex.** Is there a vertex cover of size $\leq 3$?
**Vertex cover and independent set reduce to one another**

**Theorem.** \( \text{VERTEX-COVER} \equiv_p \text{INDEPENDENT-SET}. \)

**Pf.** We show \( S \) is an independent set of size \( k \) iff \( V - S \) is a vertex cover of size \( n - k \).

\[
(G, k) \quad \quad \rightarrow \quad \quad (6, n-k)
\]

[Diagram showing a reduction process]

- \( \bullet \): independent set of size 6
- \( \circ \): vertex cover of size 4

\( \text{Yes} \rightarrow \text{Yes} \)

\( \text{No} \rightarrow \text{No} \)
Vertex cover and independent set reduce to one another

Theorem. \( \text{VERTEX-COVER} =_{P} \text{INDEPENDENT-SET} \).

Pf. We show \( S \) is an independent set of size \( k \) iff \( V - S \) is a vertex cover of size \( n - k \).

\( \Rightarrow \)

- Let \( S \) be any independent set of size \( k \).
- \( V - S \) is of size \( n - k \).
- Consider an arbitrary edge \( (u, v) \).
- \( S \) independent \( \Rightarrow \) either \( u \notin S \) or \( v \notin S \) (or both)
  \( \Rightarrow \) either \( u \in V - S \) or \( v \in V - S \) (or both).
- Thus, \( V - S \) covers \( (u, v) \).
Vertex cover and independent set reduce to one another

**Theorem.** $\text{VERTEX-COVER} \equiv_p \text{INDEPENDENT-SET}.$

**Pf.** We show $S$ is an independent set of size $k$ iff $V - S$ is a vertex cover of size $n - k$.

$\iff$

- Let $V - S$ be any vertex cover of size $n - k$.
- $S$ is of size $k$.
- Consider two nodes $u \in S$ and $v \in S$.
- Observe that $(u, v) \notin E$ since $V - S$ is a vertex cover.
- Thus, no two nodes in $S$ are joined by an edge $\Rightarrow$ $S$ independent set. $\blacksquare$
Set cover

**Set-Cover.** Given a set $U$ of elements, a collection $S$ of subsets of $U$, and an integer $k$, are there $\leq k$ of these subsets whose union is equal to $U$?

Sample application.

- $m$ available pieces of software.
- Set $U$ of $n$ capabilities that we would like our system to have.
- The $i^{th}$ piece of software provides the set $S_i \subseteq U$ of capabilities.
- Goal: achieve all $n$ capabilities using fewest pieces of software.

\[
\begin{align*}
U &= \{1, 2, 3, 4, 5, 6, 7\} \\
S_a &= \{3, 7\} & S_b &= \{2, 4\} \\
\textcolor{blue}{S_c} &= \{3, 4, 5, 6\} & \textcolor{blue}{S_d} &= \{5\} \\
S_e &= \{1\} & \textcolor{blue}{S_f} &= \{1, 2, 6, 7\} \\
k &= 2
\end{align*}
\]

a set cover instance
Vertex cover reduces to set cover

**Theorem.** \( \text{VERTEX-COVER} \leq_p \text{SET-COVER}. \)

**Pf.** Given a VERTEX-COVER instance \( G = (V, E) \) and \( k \), we construct a SET-COVER instance \((U, S)\) that has a set cover of size \( k \) iff \( G \) has a vertex cover of size \( k \).

**Construction.**
- Universe \( U = E \).
- Include one subset for each node \( v \in V \) \( : \) \( S_v = \{ e \in E : e \text{ incident to } v \} \).

![Diagram](image)

\( k = 2 \)

\[ U = \{ 1, 2, 3, 4, 5, 6, 7 \} \]
\[ S_a = \{ 3, 7 \} \]
\[ S_b = \{ 2, 4 \} \]
\[ S_c = \{ 3, 4, 5, 6 \} \]
\[ S_d = \{ 5 \} \]
\[ S_e = \{ 1 \} \]
\[ S_f = \{ 1, 2, 6, 7 \} \]

**set cover instance** \( k = 2 \)
**Vertex cover reduces to set cover**

**Lemma.** \( G = (V, E) \) contains a vertex cover of size \( k \) iff \( (U, S) \) contains a set cover of size \( k \).

**Pf.** \( \Rightarrow \) Let \( X \subseteq V \) be a vertex cover of size \( k \) in \( G \).
- Then \( Y = \{ S_v : v \in X \} \) is a set cover of size \( k \).

---

**Example:**

- **Vertex cover instance** (\( k = 2 \))
  - \( U = \{ 1, 2, 3, 4, 5, 6, 7 \} \)
  - \( S_a = \{ 3, 7 \} \)
  - \( S_b = \{ 2, 4 \} \)
  - \( S_c = \{ 3, 4, 5, 6 \} \)
  - \( S_d = \{ 5 \} \)
  - \( S_e = \{ 1 \} \)
  - \( S_f = \{ 1, 2, 6, 7 \} \)

- **Set cover instance** (\( k = 2 \)}
Vertex cover reduces to set cover

**Lemma.** $G = (V, E)$ contains a vertex cover of size $k$ iff $(U, S)$ contains a set cover of size $k$.

**Pf.** $\Leftarrow$ Let $Y \subseteq S$ be a set cover of size $k$ in $(U, S)$.
- Then $X = \{ v : S_v \in Y \}$ is a vertex cover of size $k$ in $G$. ■

---

**Vertex cover instance**  
(k = 2)

**Set cover instance**  
(k = 2)

$U = \{ 1, 2, 3, 4, 5, 6, 7 \}$  
$S_a = \{ 3, 7 \}$  
$S_b = \{ 2, 4 \}$  
$S_c = \{ 3, 4, 5, 6 \}$  
$S_d = \{ 5 \}$  
$S_e = \{ 1 \}$  
$S_f = \{ 1, 2, 6, 7 \}$
Satisfiability

Literal. A Boolean variable or its negation. \( x_i \) or \( \overline{x}_i \)

Clause. A disjunction of literals.

\[ C_j = x_1 \lor \overline{x}_2 \lor x_3 \]

Conjunctive normal form (CNF). A propositional formula \( \Phi \) that is a conjunction of clauses.

\[ \Phi = C_1 \land C_2 \land C_3 \land C_4 \]

SAT. Given a CNF formula \( \Phi \), does it have a satisfying truth assignment?

3-SAT. SAT where each clause contains exactly 3 literals (and each literal corresponds to a different variable).

\[
\Phi = \left( \overline{x}_1 \lor x_2 \lor x_3 \right) \land \left( x_1 \lor \overline{x}_2 \lor x_3 \right) \land \left( \overline{x}_1 \lor x_2 \lor x_4 \right)
\]

yes instance: \( x_1 = \text{true}, x_2 = \text{true}, x_3 = \text{false}, x_4 = \text{false} \)

Key application. Electronic design automation (EDA).
3-satisfiability reduces to independent set

Theorem. 3-SAT $\leq_p$ INDEPENDENT-SET.

Pf. Given an instance $\Phi$ of 3-SAT, we construct an instance $(G, k)$ of 
INDEPENDENT-SET that has an independent set of size $k = |\Phi|$ iff $\Phi$ is satisfiable.

Construction.

- $G$ contains 3 nodes for each clause, one for each literal.
- Connect 3 literals in a clause in a triangle.
- Connect literal to each of its negations.

$$\Phi = (\overline{x_1} \lor x_2 \lor x_3) \land (\overline{x_1} \lor x_2 \lor \overline{x_3}) \land (\overline{x_1} \lor x_2 \lor x_4)$$
3-satisfiability reduces to independent set

**Lemma.** $G$ contains independent set of size $k = |\Phi|$ iff $\Phi$ is satisfiable.

**Pf.** $\Rightarrow$ Let $S$ be independent set of size $k$.
- $S$ must contain exactly one node in each triangle.
- Set these literals to true (and remaining variables consistently).
- Truth assignment is consistent and all clauses are satisfied.

**Pf. $\Leftarrow$** Given satisfying assignment, select one true literal from each triangle. This is an independent set of size $k$. □

\[ \Phi = (\overline{x_1} \lor x_2 \lor x_3) \land (\overline{x_1} \lor \overline{x_2} \lor x_3) \land (\overline{x_1} \lor x_2 \lor \overline{x_4}) \]

Recall: $X$ reduces to $Y$ ($X \leq_p Y$) if:
- there is a polytime algorithm $f : \Sigma \to \Sigma$ such that, for every $x \in \Sigma^*$,
  - $x \in X \Rightarrow f(x) \in Y$
  - $x \notin X \Rightarrow f(x) \notin Y$

\[ k = 3 \]

Assume

\[ \text{is sat: } x_2 = 1, x_1 = 1 \]
Review

Basic reduction strategies.

- Simple equivalence: \( \text{INDEPENDENT-SET} \equiv_p \text{VERTEX-COVER} \).
- Special case to general case: \( \text{VERTEX-COVER} \leq_p \text{SET-COVER} \).
- Encoding with gadgets: \( 3\text{-SAT} \leq_p \text{INDEPENDENT-SET} \).

Transitivity. If \( X \leq_p Y \) and \( Y \leq_p Z \), then \( X \leq_p Z \).

Pf idea. Compose the two algorithms.

Ex. \( 3\text{-SAT} \leq_p \text{INDEPENDENT-SET} \leq_p \text{VERTEX-COVER} \leq_p \text{SET-COVER} \).
Search problems

**Decision problem.** Does there exist a vertex cover of size \( \leq k \)?

**Search problem.** Find a vertex cover of size \( \leq k \).

**Ex.** To find a vertex cover of size \( \leq k \):
- Determine if there exists a vertex cover of size \( \leq k \).
- Find a vertex \( v \) such that \( G - \{ v \} \) has a vertex cover of size \( \leq k - 1 \).
  (any vertex in any vertex cover of size \( \leq k \) will have this property)
- Include \( v \) in the vertex cover.
- Recursively find a vertex cover of size \( \leq k - 1 \) in \( G - \{ v \} \).

**Bottom line.** \( \text{VERTEX-COVER} =_p \text{FIND-VERTEX-COVER} \).

Optimization problems

**Decision problem.** Does there exist a vertex cover of size \( \leq k \)?

**Search problem.** Find a vertex cover of size \( \leq k \).

**Optimization problem.** Find a vertex cover of minimum size.

**Ex.** To find vertex cover of minimum size:
- (Binary) search for size \( k^* \) of min vertex cover.
- Solve corresponding search problem.

**Bottom line.** \( \text{VERTEX-COVER} =_p \text{FIND-VERTEX-COVER} =_p \text{OPTIMAL-VERTEX-COVER} \).
3-dimensional matching

**3D-Matching.** Given 3 disjoint sets $X$, $Y$, and $Z$, each of size $n$ and a set $T \subseteq X \times Y \times Z$ of triples, does there exist a set of $n$ triples in $T$ such that each element of $X \cup Y \cup Z$ is in exactly one of these triples?

$$X = \{x_1, x_2, x_3\}, \quad Y = \{y_1, y_2, y_3\}, \quad Z = \{z_1, z_2, z_3\}$$

$$T_1 = \{x_1, y_1, z_2\}, \quad T_2 = \{x_1, y_2, z_1\}, \quad T_3 = \{x_1, y_2, z_2\}$$

$$T_4 = \{x_2, y_2, z_3\}, \quad T_5 = \{x_2, y_3, z_3\}, \quad T_6 = \{x_2, y_3, z_3\}$$

$$T_7 = \{x_3, y_1, z_3\}, \quad T_8 = \{x_3, y_1, z_3\}, \quad T_9 = \{x_3, y_2, z_1\}$$

An instance of 3d-matching (with $n = 3$)

**Remark.** Generalization of bipartite matching.
3-dimensional matching

**3D-MATCHING.** Given 3 disjoint sets $X$, $Y$, and $Z$, each of size $n$ and a set $T \subseteq X \times Y \times Z$ of triples, does there exist a set of $n$ triples in $T$ such that each element of $X \cup Y \cup Z$ is in exactly one of these triples?

**Theorem.** $3$-SAT $\leq_p 3$D-MATCHING.

**Pf.** Given an instance $\Phi$ of 3-SAT, we construct an instance of 3D-MATCHING that has a perfect matching iff $\Phi$ is satisfiable.
3-colorability

3-COLOR. Given an undirected graph $G$, can the nodes be colored black, white, and blue so that no adjacent nodes have the same color?

vs.

2-COLOR (≡ $\Pi_2$ Bipartiteness) $\not\in$ P

yes instance
3-satisfiability reduces to 3-colorability

**Theorem.** \(3\text{-SAT} \leq_p 3\text{-COLOR}.\)

**Pf.** Given 3-SAT instance \(\Phi\), we construct an instance of 3-COLOR that is 3-colorable iff \(\Phi\) is satisfiable.

To show \(3\text{-COLOR}\) is \(\text{NP-complete}\):

1. Show \(3\text{-COLOR} \in \text{NP}\)
2. \(3\text{-SAT} \leq_p 3\text{-COLOR}\)

\(\text{some NP-complete}\)
3-satisfiability reduces to 3-colorability

Construction.
(i) Create a graph $G$ with a node for each literal.
(ii) Connect each literal to its negation.
(iii) Create 3 new nodes $T$, $F$, and $B$; connect them in a triangle.
(iv) Connect each literal to $B$.
(v) For each clause $C_j$, add a gadget of 6 nodes and 13 edges.

To be described later
3-satisfiability reduces to 3-colorability

**Lemma.** Graph $G$ is 3-colorable iff $\Phi$ is satisfiable.

**Pf.** $\Rightarrow$ Suppose graph $G$ is 3-colorable.

- WLOG, assume that node $T$ is colored black, $F$ is white, and $B$ is blue.
- Consider assignment that sets all black literals to true (and white to false).
- (iv) ensures each literal is colored either black or white.
- (ii) ensures that each literal is white if its negation is black (and vice versa).

\[
C_1 = x_1 \lor x_2 \lor x_3
\]

\[
C_2 = x_3 \lor \neg x_5 \lor x_7
\]
3-satisfiability reduces to 3-colorability

**Lemma.** Graph $G$ is 3-colorable iff $\Phi$ is satisfiable.

**Pf.** $\Rightarrow$ Suppose graph $G$ is 3-colorable.

- WLOG, assume that node $T$ is colored black, $F$ is white, and $B$ is blue.
- Consider assignment that sets all black literals to true (and white to false).
- (iv) ensures each literal is colored either black or white.
- (ii) ensures that each literal is white if its negation is black (and vice versa).
- (v) ensures at least one literal in each clause is black.

\[ C_j = x_1 \lor \overline{x_2} \lor x_3 \]

6-node gadget
3-satisfiability reduces to 3-colorability

**Lemma.** Graph $G$ is 3-colorable iff $\Phi$ is satisfiable.

**Pf.** $\Rightarrow$ Suppose graph $G$ is 3-colorable.

- WLOG, assume that node $T$ is colored black, $F$ is white, and $B$ is blue.
- Consider assignment that sets all black literals to true (and white to false).
- (iv) ensures each literal is colored either black or white.
- (v) ensures that each literal is white if its negation is black (and vice versa).
- (v) ensures at least one literal in each clause is black.

```latex
\begin{align*}
C_j &= x_1 \lor \overline{x_2} \lor x_3 \\
\end{align*}
```

suppose, for the sake of contradiction, that all 3 literals are white in some 3-coloring

contradiction (not a 3-coloring)
3-satisfiability reduces to 3-colorability

**Lemma.** Graph $G$ is 3-colorable iff $\Phi$ is satisfiable.

**Pf.** $\Leftarrow$ Suppose 3-SAT instance $\Phi$ is satisfiable.

- Color all true literals black and all false literals white.
- Pick one true literal; color node below that node white, and node below that blue.
- Color remaining middle row nodes blue.
- Color remaining bottom nodes black or white, as forced. ■
Polynomial-time reductions

constraint satisfaction

3-SAT

INDEPENDENT-SET
  VERTEX-COVER
  SET-COVER

DIR-HAM-CYCLE
  HAM-CYCLE
  TSP

GRAPH-3-COLOR
  PLANAR-3-COLOR

SUBSET-SUM
  SCHEDULING

packing and covering
sequencing
partitioning
numerical
Subset sum

**SUBSET-SUM.** Given natural numbers $w_1, \ldots, w_n$ and an integer $W$, is there a subset that adds up to exactly $W$?

**Ex.** $\{ 1, 4, 16, 64, 256, 1040, 1041, 1093, 1284, 1344 \}$, $W = 3754$.

**Yes.** $1 + 16 + 64 + 256 + 1040 + 1093 + 1284 = 3754$.

**Remark.** With arithmetic problems, input integers are encoded in binary. Poly-time reduction must be polynomial in binary encoding.
Subset sum

**Theorem.** 3-SAT $\leq_p$ SUBSET-SUM.

**Pf.** Given an instance $\Phi$ of 3-SAT, we construct an instance of SUBSET-SUM that has solution iff $\Phi$ is satisfiable.
3-satisfiability reduces to subset sum

**Construction.** Given 3-SAT instance $\Phi$ with $n$ variables and $k$ clauses, form $2n + 2k$ decimal integers, each of $n + k$ digits:
- Include one digit for each variable $x_i$ and for each clause $C_j$.
- Include two numbers for each variable $x_i$.
- Include two numbers for each clause $C_j$.
- Sum of each $x_i$ digit is 1;
- sum of each $C_j$ digit is 4.

**Key property.** No carries possible $\Rightarrow$
each digit yields one equation.

\[
C_1 = \neg x_1 \lor x_2 \lor x_3
\]
\[
C_2 = x_1 \lor \neg x_2 \lor x_3
\]
\[
C_3 = \neg x_1 \lor \neg x_2 \lor \neg x_3
\]

3-SAT instance

<table>
<thead>
<tr>
<th>$x_1$</th>
<th>$x_2$</th>
<th>$x_3$</th>
<th>$C_1$</th>
<th>$C_2$</th>
<th>$C_3$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>$\neg x_1$</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>$x_2$</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>$\neg x_2$</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>$x_3$</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>$\neg x_3$</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

dummies to get clause columns to sum to 4

\[
\begin{align*}
W & = 1 & 1 & 1 & 4 & 4 & 4 \\
\text{SET-SUM Instance} & & & & & & \text{111,444}
\end{align*}
\]
3-satisfiability reduces to subset sum

**Lemma.** $\Phi$ is satisfiable iff there exists a subset that sums to $W$.

**Pf.** $\Rightarrow$ Suppose $\Phi$ is satisfiable.

- Choose integers corresponding to each *true* literal.
- Since $\Phi$ is satisfiable, each $C_j$ digit sums to at least 1 from $x_i$ rows.
- Choose dummy integers to make clause digits sum to 4.

3-SAT instance

$$C_1 = \neg x_1 \lor x_2 \lor x_3$$

$$C_2 = x_1 \lor \neg x_2 \lor x_3$$

$$C_3 = \neg x_1 \lor \neg x_2 \lor \neg x_3$$

<table>
<thead>
<tr>
<th>$x_1$</th>
<th>$x_2$</th>
<th>$x_3$</th>
<th>$C_1$</th>
<th>$C_2$</th>
<th>$C_3$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>W</th>
<th>1</th>
<th>1</th>
<th>1</th>
<th>4</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>111,444</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Subset-Sum Instance
3-satisfiability reduces to subset sum

Lemma. \( \Phi \) is satisfiable iff there exists a subset that sums to \( W \).

Pf. \( \iff \) Suppose there is a subset that sums to \( W \).

\begin{itemize}
\item Digit \( x_i \) forces subset to select either row \( x_i \) or \( \neg x_i \) (but not both).
\item Digit \( C_j \) forces subset to select at least one literal in clause.
\item Assign \( x_i = \text{true} \) iff row \( x_i \) selected. \end{itemize}

\[
\begin{array}{cccccc}
 x_1 & x_2 & x_3 & C_1 & C_2 & C_3 \\
\hline
 x_1 & 1 & 0 & 0 & 0 & 1 & 0 & 100,010 \\
 \neg x_1 & 1 & 0 & 0 & 1 & 0 & 1 & 100,011 \\
 x_2 & 0 & 1 & 0 & 1 & 0 & 0 & 10,100 \\
 \neg x_2 & 0 & 1 & 0 & 0 & 1 & 1 & 10,011 \\
 x_3 & 0 & 0 & 1 & 1 & 1 & 0 & 1,110 \\
 \neg x_3 & 0 & 0 & 1 & 0 & 0 & 1 & 1,001 \\
\end{array}
\]

\begin{itemize}
\item \( C_1 = \neg x_1 \lor x_2 \lor x_3 \)
\item \( C_2 = x_1 \lor \neg x_2 \lor x_3 \)
\item \( C_3 = \neg x_1 \lor \neg x_2 \lor \neg x_3 \)
\end{itemize}

3-SAT instance

\[
\begin{aligned}
&\text{dummies to get clause columns to sum to 4} \\
&\begin{array}{cccccc}
& 0 & 0 & 0 & 0 & 1 & 0 & 0 & 100 \\
& 0 & 0 & 0 & 2 & 0 & 0 & 200 \\
& 0 & 0 & 0 & 0 & 1 & 0 & 10 \\
& 0 & 0 & 0 & 0 & 2 & 0 & 20 \\
& 0 & 0 & 0 & 0 & 0 & 1 & 1 \\
& 0 & 0 & 0 & 0 & 0 & 2 & 2 \\
\end{array}
\end{aligned}
\]

\[
\begin{array}{cccc}
 W & 1 & 1 & 4 & 4 & 4 & 111,444 \\
\end{array}
\]

Subset-Sum instance
**Partition**

**Subset-Sum.** Given natural numbers $w_1, \ldots, w_n$ and an integer $W$, is there a subset that adds up to exactly $W$?

**Partition.** Given natural numbers $v_1, \ldots, v_m$, can they be partitioned into two subsets that add up to the same value $\frac{1}{2} \sum_i v_i$?

**Theorem.** $\text{Subset-Sum} \leq_p \text{Partition}.$

**Pf.** Let $W, w_1, \ldots, w_n$ be an instance of $\text{Subset-Sum}$.

- Create instance of $\text{Partition}$ with $m = n + 2$ elements.
  - $v_1 = w_1, v_2 = w_2, \ldots, v_n = w_n, \quad v_{n+1} = 2 \sum_i w_i - W, \quad v_{n+2} = \sum_i w_i + W$

- Lemma: there exists a subset that sums to $W$ iff there exists a partition since elements $v_{n+1}$ and $v_{n+2}$ cannot be in the same partition. •

\[
\begin{align*}
\text{subset A} & \quad v_{n+1} = 2 \sum_i w_i - W & W \\
\text{subset B} & \quad v_{n+2} = \sum_i w_i + W & \sum_i w_i - W
\end{align*}
\]
Polynomial-time reductions

constraint satisfaction

3-SAT

INDEPENDENT-SET

DIR-HAM-CYCLE

GRAPH-3-COLOR

SUBSET-SUM

VERTEX-COVER

HAM-CYCLE

PLANAR-3-COLOR

SCHEDULING

SET-COVER

TSP

packing and covering

sequencing

partitioning

numerical
Karp's 21 NP-complete problems

FIGURE 1 - Complete Problems

Dick Karp (1972)
1985 Turing Award