8. Intractability I

- poly-time reductions
- packing and covering problems
- constraint satisfaction problems
- sequencing problems
- partitioning problems
- graph coloring
- numerical problems
8. Intractability I

- poly-time reductions
- packing and covering problems
- constraint satisfaction problems
- sequencing problems
- partitioning problems
- graph coloring
- numerical problems
Algorithm design patterns and antipatterns

Algorithm design patterns.
- Greedy.
- Divide and conquer.
- Dynamic programming.
- Duality.
- Reductions.
- Local search.
- Randomization.

Algorithm design antipatterns.
- **NP-completeness.** $O(n^k)$ algorithm unlikely.
- **PSPACE-completeness.** $O(n^k)$ certification algorithm unlikely.
- Undecidability. No algorithm possible.
Classify problems according to computational requirements

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

A working definition. Those with poly-time algorithms.


Turing machine, word RAM, uniform circuits, ...

Theory. Definition is broad and robust.

Practice. Poly-time algorithms scale to huge problems.

constants tend to be small, e.g., $3n^2$
Classify problems according to computational requirements

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

A working definition. Those with poly-time algorithms.

<table>
<thead>
<tr>
<th>yes</th>
<th>probably no</th>
</tr>
</thead>
<tbody>
<tr>
<td>shortest path</td>
<td>longest path</td>
</tr>
<tr>
<td>min cut</td>
<td>max cut</td>
</tr>
<tr>
<td>2-satisfiability</td>
<td>3-satisfiability</td>
</tr>
<tr>
<td>planar 4-colorability</td>
<td>planar 3-colorability</td>
</tr>
<tr>
<td>bipartite vertex cover</td>
<td>vertex cover</td>
</tr>
<tr>
<td>matching</td>
<td>3d-matching</td>
</tr>
<tr>
<td>primality testing</td>
<td>factoring</td>
</tr>
<tr>
<td>linear programming</td>
<td>integer linear programming</td>
</tr>
</tbody>
</table>
Classify problems

Desiderata. Classify problems according to those that can be solved in polynomial time and those that cannot.

Provably requires exponential time.
- Given a constant-size program, does it halt in at most $k$ steps?
- Given a board position in an $n$-by-$n$ generalization of checkers, can black guarantee a win?

Frustrating news. Huge number of fundamental problems have defied classification for decades.
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 $Y$.
Poly-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 of $Y$ sent to oracle $\Rightarrow$ instances of $Y$ must be of polynomial size.

Novice mistake. Confusing $X \leq_p Y$ with $Y \leq_p X$. 
Polynomial transformations

**Def.** Problem $X$ polynomial (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 $Y$.

**Def.** Problem $X$ polynomial (Karp) transforms to problem $Y$ if given any instance $x$ of $X$, we can construct an instance $y$ of $Y$ such that $x$ is a yes instance of $X$ iff $y$ is a yes instance of $Y$.

we require $|y|$ to be of size polynomial in $|x|$.

**Note.** Polynomial transformation is polynomial reduction with just one call to oracle for $Y$, exactly at the end of the algorithm for $X$. Almost all previous reductions were of this form.

**Open question.** Are these two concepts the same with respect to $\textbf{NP}$? we abuse notation $\leq_p$ and blur distinction.
Intractability: quiz 1

Suppose that $X \leq_p Y$. Which of the following can we infer?

A. If $X$ can be solved in polynomial time, then so can $Y$.
B. $X$ can be solved in poly time iff $Y$ can be solved in poly time.
C. If $X$ cannot be solved in polynomial time, then neither can $Y$.
D. If $Y$ cannot be solved in polynomial time, then neither can $X$. 
Which of the following poly-time reductions are known?

A. **FIND-MAX-FLOW \leq_p FIND-MIN-CUT.**

B. **FIND-MIN-CUT \leq_p FIND-MAX-FLOW.**

C. Both A and B.

D. Neither A nor B.
Poly-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 $X \equiv_p Y$. In this case, $X$ can be solved in polynomial time iff $Y$ can be.

**Bottom line.** Reductions classify problems according to relative difficulty.
8. **Intractability I**

- poly-time reductions
- packing and covering problems
- constraint satisfaction problems
- sequencing problems
- partitioning problems
- graph coloring
- numerical problems
Independent set

**INDEPENDENT-SET.** Given a graph $G = (V, E)$ and an integer $k$, is there a subset of $k$ (or more) vertices such that no two are adjacent?

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

**VERTEX-COVER.** Given a graph $G = (V, E)$ and an integer $k$, is there a subset of $k$ (or fewer) vertices such that each edge is incident to at least one vertex in the subset?

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

**Ex.** Is there a vertex cover of size $\leq 3$?

![Diagram of a graph with vertex cover and independent set markings]
Consider the following graph $G$. Which are true?

A. The white vertices are a vertex cover of size 7.
B. The black vertices are an independent set of size 3.
C. Both A and B.
D. Neither A nor B.
Theorem. \textsc{Independent-Set} $\equiv_p \textsc{Vertex-Cover}$.

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

![Graph diagram with vertices and edges, indicating an independent set of size 6 and a vertex cover of size 4.]
Vertex cover and independent set reduce to one another

**Theorem.** \textsc{Independent-Set} \( \equiv_p \textsc{Vertex-Cover} \).

**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 \((u, v) \in E\).
- \( S \) 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 \((u, v)\). \( \blacksquare \)
Vertex cover and independent set reduce to one another

**Theorem.** \textsc{Independent-Set} $\equiv_p \textsc{Vertex-Cover}$.

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

$\Leftarrow$

- Let $V - S$ be any vertex cover of size $n - k$.
- $S$ is of size $k$.
- Consider an arbitrary edge $(u, v) \in E$.
- $V - S$ is a vertex cover $\Rightarrow$ either $u \in V - S$, or $v \in V - S$, or both.
  $\Rightarrow$ either $u \notin S$, or $v \notin S$, or both.
- Thus, $S$ is an independent set. \hfill $\blacksquare$
**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.

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

**a set cover instance**
Given the universe \( U = \{ 1, 2, 3, 4, 5, 6, 7 \} \) and the following sets, which is the minimum size of a set cover?

A. 1

\[ U = \{ 1, 2, 3, 4, 5, 6, 7 \} \]

\[ S_a = \{ 1, 4, 6 \} \quad S_b = \{ 1, 6, 7 \} \]

\[ S_c = \{ 1, 2, 3, 6 \} \quad S_d = \{ 1, 3, 5, 7 \} \]

\[ S_e = \{ 2, 6, 7 \} \quad S_f = \{ 3, 4, 5 \} \]

B. 2

C. 3

D. None of the above.
Vertex cover reduces to set cover

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

**Pf.** Given a \( \text{VERTEX-COVER} \) instance \( G = (V, E) \) and \( k \), we construct a \( \text{SET-COVER} \) instance \( (U, S, k) \) 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 \( v \in V : S_v = \{ e \in E : e \text{ incident to } v \} \).

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

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

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

“yes” instances of VERTEX-COVER are solved correctly

![Graph](image)

vertex cover instance (k = 2)

set cover instance (k = 2)

$U = \{ 1, 2, 3, 4, 5, 6, 7 \}$

$S_a = \{ 3, 7 \}$

$S_b = \{ 2, 4 \}$

$\underline{S_c} = \{ 3, 4, 5, 6 \}$

$S_d = \{ 5 \}$

$S_e = \{ 1 \}$

$\underline{S_f} = \{ 1, 2, 6, 7 \}$
Vertex cover reduces to set cover

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

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

<table>
<thead>
<tr>
<th>vertex cover instance</th>
<th>set cover instance</th>
</tr>
</thead>
<tbody>
<tr>
<td>( k = 2 )</td>
<td>( k = 2 )</td>
</tr>
</tbody>
</table>

**Example:**
- **Graph:**
  - \( U = \{ 1, 2, 3, 4, 5, 6, 7 \} \)
  - \( S_a = \{ 3, 7 \} \)
  - \( S_b = \{ 2, 4 \} \)
  - \( S_c = \{ 3, 4, 5, 6 \} \)
  - \( S_d = \{ 5 \} \)
  - \( S_e = \{ 1 \} \)
  - \( S_f = \{ 1, 2, 6, 7 \} \)

"no" instances of **VERTEX-COVER** are solved correctly
8. Intractability I

- poly-time reductions
- packing and covering problems
- constraint satisfaction problems
- sequencing problems
- partitioning problems
- graph coloring
- numerical problems
Satisfiability

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

**Clause.** A disjunction of literals. \( C_j = x_1 \vee \overline{x_2} \vee x_3 \)

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

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

**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).

\[ \Phi = (\overline{x_1} \vee x_2 \vee x_3) \wedge (x_1 \vee \overline{x_2} \vee x_3) \wedge (\overline{x_1} \vee x_2 \vee x_4) \]

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

**Key application.** Electronic design automation (EDA).
Satisfiability is hard

**Scientific hypothesis.** There does not exists a poly-time algorithm for 3-SAT.

**P vs. NP.** This hypothesis is equivalent to $P \neq NP$ conjecture.

---

Computer Scientists have so much funding and time and can't even figure out the boolean satisfiability problem. SAT!

[https://www.facebook.com/pg/npcompletesteen](https://www.facebook.com/pg/npcompletesteen)
Theorem. 3-SAT \leq_p \text{INDEPENDENT-SET}.

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

\textbf{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.

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

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

**Pf.** \( \Rightarrow \) Consider any satisfying assignment for \( \Phi \).

- Select one true literal from each clause/triangle.
- This is an independent set of size \( k = |\Phi| \).  

\[
\begin{align*}
\Phi &= (\overline{x_1} \lor x_2 \lor x_3) \land (x_1 \lor \overline{x_2} \lor x_3) \land (\overline{x_1} \lor x_2 \lor x_4)
\end{align*}
\]
3-satisfiability reduces to independent set

Lemma. \( \Phi \) is satisfiable iff \( G \) contains an independent set of size \( k = |\Phi| \).

Pf. \( \iff \) Let \( S \) be independent set of size \( k \).

- \( S \) must contain exactly one node in each triangle.
- Set these literals to \textit{true} (and remaining literals consistently).
- All clauses in \( \Phi \) are satisfied. \( \blacksquare \)

\[
\begin{align*}
\Phi & = \left( \overline{x_1} \lor x_2 \lor x_3 \right) \land \left( x_1 \lor \overline{x_2} \lor x_3 \right) \land \left( \overline{x_1} \lor x_2 \lor x_4 \right)
\end{align*}
\]
Basic reduction strategies.

- Simple equivalence: \textsc{Independent-Set} \equiv_p \textsc{Vertex-Cover}.
- Special case to general case: \textsc{Vertex-Cover} \leq_p \textsc{Set-Cover}.
- Encoding with gadgets: 3-SAT \leq_p \textsc{Independent-Set}.

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

Pf idea. Compose the two algorithms.

\textbf{Ex.} 3-SAT \leq_p \textsc{Independent-Set} \leq_p \textsc{Vertex-Cover} \leq_p \textsc{Set-Cover}.
**Decision, search, and 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.

**Goal.** Show that all three problems poly-time reduce to one another.
8. **Intractability I**

- poly-time reductions
- packing and covering problems
- constraint satisfaction problems
- sequencing problems
- partitioning problems
- graph coloring
- numerical problems
**Hamilton cycle**

**HAMILTON-CYCLE.** Given an undirected graph $G = (V, E)$, does there exist a cycle $\Gamma$ that visits every node exactly once?
**Hamilton cycle**

**Hamilton-Cycle.** Given an undirected graph $G = (V, E)$, does there exist a cycle $\Gamma$ that visits every node exactly once?

```
no
```
Directed Hamilton cycle reduces to Hamilton cycle

**Directed-Hamilton-Cycle.** Given a directed graph $G = (V, E)$, does there exist a directed cycle $\Gamma$ that visits every node exactly once?

**Theorem.** Directed-Hamilton-Cycle $\leq_p$ Hamilton-Cycle.

**Pf.** Given a directed graph $G = (V, E)$, construct a graph $G'$ with $3n$ nodes.

![Directed graph G](image1)

![Undirected graph G'](image2)
Directed Hamilton cycle reduces to Hamilton cycle

Lemma. $G$ has a directed Hamilton cycle iff $G'$ has a Hamilton cycle.

Pf. $\implies$

- Suppose $G$ has a directed Hamilton cycle $\Gamma$.
- Then $G'$ has an undirected Hamilton cycle (same order). □

Pf. $\impliedby$

- Suppose $G'$ has an undirected Hamilton cycle $\Gamma'$.
- $\Gamma'$ must visit nodes in $G'$ using one of following two orders:
  - $\ldots, \text{black}, \text{white}, \text{blue}, \text{black}, \text{white}, \text{blue}, \text{black}, \ldots$
  - $\ldots, \text{black}, \text{blue}, \text{white}, \text{black}, \text{blue}, \text{white}, \text{black}, \text{blue}, \text{white}, \ldots$
- Black nodes in $\Gamma'$ comprise either a directed Hamilton cycle $\Gamma$ in $G$, or reverse of one. □
3-satisfiability reduces to directed Hamilton cycle

**Theorem.** $3$-$\text{Sat} \leq_p \text{DIRECTED-HAMILTON-CYCLE}$. 

**Pf.** Given an instance $\Phi$ of $3$-$\text{Sat}$, we construct an instance $G$ of $\text{DIRECTED-HAMILTON-CYCLE}$ that has a Hamilton cycle iff $\Phi$ is satisfiable.

**Construction overview.** Let $n$ denote the number of variables in $\Phi$. We will construct a graph $G$ that has $2^n$ Hamilton cycles, with each cycle corresponding to one of the $2^n$ possible truth assignments.
3-satisfiability reduces to directed Hamilton cycle

**Construction.** Given 3-SAT instance $\Phi$ with $n$ variables $x_i$ and $k$ clauses.
- Construct $G$ to have $2^n$ Hamilton cycles.
- Intuition: traverse path $i$ from left to right $\Leftrightarrow$ set variable $x_i = true$. 
Which is truth assignment corresponding to Hamilton cycle below?

A. $x_1 = true, x_2 = true, x_3 = true$

B. $x_1 = true, x_2 = true, x_3 = false$

C. $x_1 = false, x_2 = false, x_3 = true$

D. $x_1 = false, x_2 = false, x_3 = false$
3-satisfiability reduces to directed Hamilton cycle

Construction. Given 3-SAT instance $\Phi$ with $n$ variables $x_i$ and $k$ clauses.
  - For each clause: add a node and 2 edges per literal.
3-satisfiability reduces to directed Hamilton cycle

Construction. Given 3-SAT instance $\Phi$ with $n$ variables $x_i$ and $k$ clauses.
- For each clause: add a node and 2 edges per literal.

$$C_1 = x_1 \lor \overline{x_2} \lor x_3$$  clause node 1

$$C_2 = \overline{x_1} \lor \overline{x_2} \lor \overline{x_3}$$  clause node 2

\[ C_1 = x_1 \lor \overline{x_2} \lor x_3 \]

\[ C_2 = \overline{x_1} \lor \overline{x_2} \lor \overline{x_3} \]
3-satisfiability reduces to directed Hamilton cycle

Lemma.  \( \Phi \) is satisfiable iff \( G \) has a Hamilton cycle.

Pf.  \( \Rightarrow \)

- Suppose 3-SAT instance \( \Phi \) has satisfying assignment \( x^* \).
- Then, define Hamilton cycle \( \Gamma \) in \( G \) as follows:
  - if \( x_i^* = true \), traverse row \( i \) from left to right
  - if \( x_i^* = false \), traverse row \( i \) from right to left
  - for each clause \( C_j \), there will be at least one row \( i \) in which we are going in “correct” direction to splice clause node \( C_j \) into cycle
  (and we splice in \( C_j \) exactly once)  \( \blacksquare \)
3-satisfiability reduces to directed Hamilton cycle

Lemma. \( \Phi \) is satisfiable iff \( G \) has a Hamilton cycle.

Pf. \( \iff \)

- Suppose \( G \) has a Hamilton cycle \( \Gamma \).
- If \( \Gamma \) enters clause node \( C_j \), it must depart on mate edge.
  - nodes immediately before and after \( C_j \) are connected by an edge \( e \in E \)
  - removing \( C_j \) from cycle, and replacing it with edge \( e \) yields Hamilton cycle on \( G – \{ C_j \} \)
- Continuing in this way, we are left with a Hamilton cycle \( \Gamma' \) in \( G – \{ C_1 , C_2 , \ldots , C_k \} \).
- Set \( x_i^* = true \) if \( \Gamma' \) traverses row \( i \) left-to-right; otherwise, set \( x_i^* = false \).
- traversed in “correct” direction, and each clause is satisfied. \( \blacksquare \)
Poly-time reductions

- constraint satisfaction
  - 3-Sat
    - 3-SAT poly-time reduces to INDEPENDENT-SET
    - INDEPENDENT-SET
      - VERTEX-COVER
        - SET-COVER
    - DIR-HAM-CYCLE
      - HAM-CYCLE
    - 3-COLOR
    - SUBSET-SUM
      - KNAPSACK

packing and covering  sequencing  partitioning  numerical
8. INTRACTABILITY I

- poly-time reductions
- packing and covering problems
- constraint satisfaction problems
- sequencing problems
- partitioning problems
- graph coloring
- numerical problems
My hobby

NP–Complete by Randall Munro
http://xkcd.com/287
Creative Commons Attribution–NonCommercial 2.5
Subset sum

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

**Ex.**  \{ 215, 215, 275, 275, 355, 355, 420, 420, 580, 580, 655, 655 \},  \quad W = 1505.

**Yes.**  $215 + 355 + 355 + 580 = 1505$.

**Remark.** With arithmetic problems, input integers are encoded in binary. Poly-time reduction must be polynomial in binary encoding.
**Theorem.** \(3\text{-SAT} \leq_p \text{SUBSET-SUM}.\)

**Pf.** Given an instance \(\Phi\) of 3-SAT, we construct an instance of \text{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 having $n + k$ digits:

- Include one digit for each variable $x_i$ and one digit 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.

\[
\begin{align*}
C_1 &= \neg x_1 \lor x_2 \lor x_3 \\
C_2 &= x_1 \lor \neg x_2 \lor x_3 \\
C_3 &= \neg x_1 \lor \neg x_2 \lor \neg x_3
\end{align*}
\]

3-SAT instance
3-satisfiability reduces to subset sum

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

Pf. $\Rightarrow$ Suppose 3-SAT instance $\Phi$ has satisfying assignment $x^*$.

- If $x^*_i = \text{true}$, select integer in row $x_i$;
  otherwise, select integer in row $\neg x_i$.
- Each $x_i$ digit sums to 1.
- Since $\Phi$ is satisfiable, each $C_j$ digit sums to at least 1 from $x_i$ and $\neg x_i$ rows.
- Select dummy integers to make $C_j$ digits sum to 4. •

$$C_1 = \neg x_1 \lor x_2 \lor x_3$$
$$C_2 = x_1 \lor \neg x_2 \lor x_3$$
$$C_3 = \neg x_1 \lor \neg x_2 \lor \neg x_3$$
3-satisfiability reduces to subset sum

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

**Pf.** $\iff$ Suppose there exists a subset $S^*$ that sums to $W$.

- Digit $x_i$ forces subset $S^*$ to select either row $x_i$ or row $\neg x_i$ (but not both).
- If row $x_i$ selected, assign $x_i^* = true$; otherwise, assign $x_i^* = false$.

Digit $C_j$ forces subset $S^*$ to select at least one literal in clause. ■

<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>

| dummies to get clause columns to sum to 4 |
|------|------|------|------|------|
| 0    | 0    | 0    | 1    | 0    | 0    |
| 0    | 0    | 0    | 2    | 0    | 0    |
| 0    | 0    | 0    | 0    | 1    | 0    |
| 0    | 0    | 0    | 0    | 2    | 0    |
| 0    | 0    | 0    | 0    | 0    | 1    |
| 0    | 0    | 0    | 0    | 0    | 2    |

| $W$ | 1 | 1 | 1 | 4 | 4 | 4 | 111,444 |

3-SAT instance

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

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

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

SUBSET-SUM instance
**Subset Sum reduces to Knapsack**

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

**Knapsack.** Given a set of items $X$, weights $u_i \geq 0$, values $v_i \geq 0$, a weight limit $U$, and a target value $V$, is there a subset $S \subseteq X$ such that:

$$\sum_{i \in S} u_i \leq U, \quad \sum_{i \in S} v_i \geq V$$

Recall. $O(n U)$ dynamic programming algorithm for Knapsack.

**Challenge.** Prove Subset-Sum $\leq_P$ Knapsack.

**Pf.** Given instance $(w_1, \ldots, w_n, W)$ of Subset-Sum, create Knapsack instance:
Poly-time reductions

3-SAT poly-time reduces to INDEPENDENT-SET

INDEPENDENT-SET

VERTEX-COVER

SET-COVER

DIR-HAM-CYCLE

HAM-CYCLE

3-COLOR

SUBSET-SUM

KNAPSACK

constraint satisfaction

packing and covering

sequencing

partitioning

numerical
Karp’s 20 poly-time reductions from satisfiability

FIGURE 1 - Complete Problems

Dick Karp (1972)
1985 Turing Award