8. **INTRACTABILITY**

- 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. \(O(n^a)\) algorithm unlikely.
- PSPACE-completeness. \(O(n^b)\) 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 polynomial-time algorithms.

**Theory.** Definition is broad and robust.

**Practice.** Poly-time algorithms scale to huge problems.
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.

<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>

Polynomial-time reductions

Desiderata'. Suppose we could solve $X$ 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 $Y \leq_P X$. 

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.
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 \( X \equiv_p Y \). In this case, \( X \) can be solved in polynomial time iff \( Y \) can be.

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

### 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 \( \geq 7 \)?

### 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. VERTEX-COVER \( \equiv_p \) INDEPENDENT-SET.

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

\[ \implies \]

\begin{itemize}
  \item Let \( V - S \) be any vertex cover of size \( n - k \).
  \item \( S \) is of size \( k \).
  \item Consider two nodes \( u \in S \) and \( v \in S \).
  \item Observe that \( (u, v) \notin E \) since \( V - S \) is a vertex cover.
  \item Thus, no two nodes in \( S \) are joined by an edge \( \implies S \) independent set.
\end{itemize}

\[ \Leftarrow \]

\begin{itemize}
  \item Let \( V - S \) be any vertex cover of size \( n - k \).
  \item \( S \) is of size \( k \).
  \item Consider two nodes \( u \in S \) and \( v \in S \).
  \item Observe that \( (u, v) \notin E \) since \( V - S \) is a vertex cover.
  \item Thus, no two nodes in \( S \) are joined by an edge \( \implies S \) independent set.
\end{itemize}

Set cover

\textbf{Set-Cover.} Given a set \( U \) of elements, a collection \( S_1, S_2, \ldots, S_m \) of subsets of \( U \), and an integer \( k \), does there exist a collection of \( \leq k \) of these sets whose union is equal to \( U \)?

Sample application.

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

\begin{center}
\begin{tabular}{c c c}
\hline
\( U \) & \( = \{ 1, 2, 3, 4, 5, 6, 7 \} \) \\
\( S_1 \) & \( = \{ 3, 7 \} \) & \( S_4 \) & \( = \{ 2, 4 \} \) \\
\( S_2 \) & \( = \{ 3, 4, 5, 6 \} \) & \( S_5 \) & \( = \{ 5 \} \) \\
\( S_3 \) & \( = \{ 1 \} \) & \( S_6 \) & \( = \{ 1, 2, 6, 7 \} \) \\
\( k \) & \( = 2 \) \\
\hline
\end{tabular}
\end{center}

\textit{a set cover instance}
Theorem. \( \text{VERTEX-COVER} \leq_p \text{SET-COVER} \).

Pf. Given a \( \text{VERTEX-COVER} \) instance \( G = (V, E) \), we construct a \( \text{SET-COVER} \) 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 set for each node \( v \in V \): \( S_v = \{ e \in E : e \text{ incident to } v \} \).

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

Pf. \( \Leftarrow \) 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 \( k \).

\( \Rightarrow \) 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 \( G \).
Satisfiability

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

Clause. A disjunction of literals. \( C_j = x_1 \lor x_2 \lor x_3 \).

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

\( \Phi = C_1 \land C_2 \land C_3 \land C_4 \)

SAT. Given 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).

\( \Phi = (x_1 \lor x_2 \lor x_3) \land (x_1 \lor \overline{x}_2 \lor x_3) \land (x_1 \lor x_2 \lor \overline{x}_3) \)

Satisfiable \( \iff \) \( \Phi \) is satisfiable.

Key application. Electronic design automation (EDA).

3-satisfiability reduces to independent set

Lemma. \( G \) contains independent set of size \( k = |\Phi| \) if \& only if \( \Phi \) is satisfiable.

\[ \begin{align*} &G \text{ contains independent set of size } k = |\Phi| \iff \Phi \text{ is satisfiable.} \\
\]

Pf. \( \Rightarrow \) Let \( S \) be independent set of size \( k \).
\( \bullet \) \( S \) must contain exactly one node in each triangle.
\( \bullet \) \( S \) must contain exactly one node in each triangle.
\( \bullet \) Set these literals to \( \text{true} \) (and remaining variables consistently).
\( \bullet \) 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 \( k \).

\[ \begin{align*} &\Phi = (x_1 \lor x_2 \lor x_3) \land (x_1 \lor \overline{x}_2 \lor x_3) \land (x_1 \lor x_2 \lor \overline{x}_3) \\
\]

3-satisfiability reduces to independent set

Review

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

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

Pf idea. Compose the two algorithms.

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

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. \( \text{VERTEX-COVER} \equiv_p \text{FIND-VERTEX-COVER} \equiv_p \text{OPTIMAL-VERTEX-COVER} \).

8. INTRACTABILITY I

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

Hamilton cycle

\text{HAM-CYCLE}. Given an undirected graph \( G = (V, E) \), does there exist a simple cycle \( \Gamma \) that contains every node in \( V \)?
**Hamilton cycle**

**HAM-CYCLE.** Given an undirected graph $G = (V, E)$, does there exist a simple cycle $\Gamma$ that contains every node in $V$?

**Directed hamilton cycle reduces to hamilton cycle**

**Directed hamilton cycle reduces to hamilton cycle**

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

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

**Pf.** $\Leftarrow$
- Suppose $G'$ has an undirected Hamilton cycle $\Gamma'$.
- $\Gamma'$ must visit nodes in $G'$ using one of following two orders:
  - $\ldots, B, G, R, B, G, R, B, G, R, B, \ldots$
- Blue nodes in $\Gamma'$ make up directed Hamilton cycle $\Gamma$ in $G$, or reverse of one.

**3-satisfiability reduces to directed hamilton cycle**

**Theorem.** 3-SAT $\leq_p$ DIR-HAM-CYCLE.

**Pf.** Given an instance $\Phi$ of 3-SAT, we construct an instance of DIR-HAM-CYCLE that has a Hamilton cycle iff $\Phi$ is satisfiable.

**Construction.** First, create graph that has $2^n$ Hamilton cycles which correspond in a natural way to $2^n$ 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 $\iff$ set variable $x_i = true$.

**Lemma.** $\Phi$ is satisfiable iff $G$ has a Hamilton cycle.

**Pf.** $\implies$
- Suppose 3-SAT instance has satisfying assignment $x^*$.  
- Then, define Hamilton cycle 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)

**Pf.** $\impliedby$
- 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 $x_i^* = true$ iff $\Gamma'$ traverses row $i$ left to right.
- Since $\Gamma$ visits each clause node $C_j$, at least one of the paths is traversed in "correct" direction, and each clause is satisfied. $\blacksquare$
3-satisfiability reduces to longest path

**LONGEST-PATH.** Given a directed graph $G = (V, E)$, does there exist a simple path consisting of at least $k$ edges?

**Theorem.** $3$-$\text{Sat} \leq_p \text{LONGEST-PATH}$. 

**Pf 1.** Redo proof for $\text{DIR-HAM-CYCLE}$, ignoring back-edge from $r$ to $s$.  
**Pf 2.** Show $\text{HAM-CYCLE} \leq_p \text{LONGEST-PATH}$. 

Traveling salesperson problem

**TSP.** Given a set of $n$ cities and a pairwise distance function $d(u, v)$, is there a tour of length $\leq D$?

Optimal TSP tour  
http://www.tsp.gatech.edu
Traveling salesperson problem

TSP. Given a set of \( n \) cities and a pairwise distance function \( d(u,v) \), is there a tour of length \( \leq D \)?

Hamilton cycle reduces to traveling salesperson problem

TSP. Given a set of \( n \) cities and a pairwise distance function \( d(u,v) \), is there a tour of length \( \leq D \)?

HAM-CYCLE. Given an undirected graph \( G = (V,E) \), does there exist a simple cycle \( \Gamma \) that contains every node in \( V \)?

Theorem. HAM-CYCLE \( \leq_p \) TSP.

Pf.

- Given instance \( G = (V,E) \) of HAM-CYCLE, create \( n \) cities with distance function
  \[
  d(u,v) = \begin{cases} 
  1 & \text{if } (u,v) \in E \\
  2 & \text{if } (u,v) \notin E
  \end{cases}
  \]

- TSP instance has tour of length \( \leq n \) iff \( G \) has a Hamilton cycle.

Remark. TSP instance satisfies triangle inequality: \( d(u,w) \leq d(u,v) + d(v,w) \).

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

8. INTRACTABILITY I

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

SECTION 8.6
3-dimensional matching

**3D-MATCHING.** Given $n$ instructors, $n$ courses, and $n$ times, and a list of the possible courses and times each instructor is willing to teach, is it possible to make an assignment so that all courses are taught at different times?

<table>
<thead>
<tr>
<th>instructor</th>
<th>course</th>
<th>time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Wayne</td>
<td>COS 226</td>
<td>TTh 11-12:20</td>
</tr>
<tr>
<td>Wayne</td>
<td>COS 423</td>
<td>MW 11-12:20</td>
</tr>
<tr>
<td>Wayne</td>
<td>COS 423</td>
<td>TTh 11-12:20</td>
</tr>
<tr>
<td>Tardos</td>
<td>COS 423</td>
<td>TTh 3-4:20</td>
</tr>
<tr>
<td>Kleinberg</td>
<td>COS 226</td>
<td>TTh 3-4:20</td>
</tr>
<tr>
<td>Kleinberg</td>
<td>COS 226</td>
<td>MW 11-12:20</td>
</tr>
<tr>
<td>Kleinberg</td>
<td>COS 423</td>
<td>MW 11-12:20</td>
</tr>
</tbody>
</table>

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?

\[
X = \{ x_1, x_2, x_3 \}, \quad Y = \{ y_1, y_2, y_3 \}, \quad Z = \{ z_1, z_2, z_3 \} \\
T_1 = \{ x_1, y_1, z_2 \}, \quad T_2 = \{ x_1, y_2, z_1 \}, \quad T_3 = \{ x_1, y_2, z_1 \} \\
T_4 = \{ x_2, y_2, z_3 \}, \quad T_5 = \{ x_2, y_3, z_1 \}, \quad T_7 = \{ x_3, y_1, z_3 \} \\
T_8 = \{ x_3, y_1, z_1 \}, \quad T_9 = \{ x_3, y_2, z_1 \} \\
\]

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

**Remark.** Generalization of bipartite matching.

3-satisfiability reduces to 3-dimensional matching

**Construction.** (part 1)
- Create gadget for each variable $x_i$ with $2k$ core elements and $2k$ tip ones.
- Theorem. **3-SAT \( \leq_p \) 3D-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.
- **Number of clauses**
- **Core elements**
- **Clause 1 tips**
- **Clause 2 tips**
- **Clause 3 tips**
- **A gadget for variable $x_i$ ($k = 4$)
**3-satisfiability reduces to 3-dimensional matching**

**Construction.** (part 1)
- Create gadget for each variable $x_i$ with $2k$ core elements and $2k$ tip ones.
- No other triples will use core elements.
- In gadget for $x_i$, any perfect matching must use either all gray triples (corresponding to $x_i = \text{true}$) or all blue ones (corresponding to $x_i = \text{false}$).

**Construction.** (part 2)
- Create gadget for each clause $C_j$ with two elements and three triples.
- Exactly one of these triples will be used in any 3d-matching.
- Ensures any perfect matching uses either (i) grey core of $x_1$ or
(ii) blue core of $x_2$ or (iii) grey core of $x_3$.

**Construction.** (part 3)
- There are $2nk$ tips: $nk$ covered by blue/grey triples; $k$ by clause triples.
- To cover remaining $(n - 1)k$ tips, create $(n - 1)k$ cleanup gadgets:
same as clause gadget but with $2nk$ triples, connected to every tip.

**Lemma.** Instance $(X, Y, Z)$ has a perfect matching iff $\Phi$ is satisfiable.

Q. What are $X$, $Y$, and $Z$?
Lemma. Instance \((X, Y, Z)\) has a perfect matching iff \(\Phi\) is satisfiable.

Q. What are \(X\), \(Y\), and \(Z\)?
A. \(X = \text{red}, Y = \text{green}, \text{ and } Z = \text{blue}\).

Pf. \(\Rightarrow\) If 3d-matching, then assign \(x_i\) according to gadget \(x_i\).

Pf. \(\Leftarrow\) If \(\Phi\) is satisfiable, use any true literal in \(C_j\) to select gadget \(C_j\) triple. \(\blacksquare\)

3-colorability

Given an undirected graph \(G\), can the nodes be colored red, green, and blue so that no adjacent nodes have the same color?
**Application: register allocation**

**Register allocation.** Assign program variables to machine register so that no more than \( k \) registers are used and no two program variables that are needed at the same time are assigned to the same register.

**Interference graph.** Nodes are program variables names; edge between \( u \) and \( v \) if there exists an operation where both \( u \) and \( v \) are "live" at the same time.

**Observation.** [Chaitin 1982] Can solve register allocation problem iff interference graph is \( k \)-colorable.

**Fact.** \( \text{3-COLOR} \leq_p \text{K-REGISTER-ALLOCATION} \) for any constant \( k \geq 3 \).

---

**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.

---

**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.

---

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

**Pf.**  
- Suppose graph \( G \) is 3-colorable.
  - Consider assignment that sets all \( T \) literals to true.
  - (iv) ensures each literal is \( T \) or \( F \).
  - (ii) ensures a literal and its negation are opposites.

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

**Pf.**  

**⇒** Suppose graph $G$ is 3-colorable.
- Consider assignment that sets all $T$ literals to true.
- (iv) ensures each literal is $T$ or $F$.
- (ii) ensures a literal and its negation are opposites.
- (v) ensures at least one literal in each clause is $T$.

**⇐** Suppose 3-SAT instance $\Phi$ is satisfiable.
- Color all true literals $T$.
- Color node below green node $F$, and node below that $B$.
- Color remaining middle row nodes $B$.
- Color remaining bottom nodes $T$ or $F$ as forced.  

---

**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

---

**3-satisfiability reduces to 3-colorability**

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

**Pf.**  

**⇒** Suppose graph $G$ is 3-colorable.
- Consider assignment that sets all $T$ literals to true.

**⇐** Suppose 3-SAT instance $\Phi$ is satisfiable.
- Consider assignment that sets all $T$ literals to true.
8. INTRACTABILITY

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

Subset sum

Theorem. 3-SAT \leq_p \text{SUBSET-SUM}.

Proof. Given an instance \( \Phi \) of 3-SAT, we construct an instance of \text{SUBSET-SUM} that has solution if \( \Phi \) 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.

\[
\begin{align*}
C_1 &= \neg x_1 \vee x_2 \vee x_3 \\
C_2 &= x_1 \vee \neg x_2 \vee x_3 \\
C_3 &= \neg x_1 \vee \neg x_2 \vee \neg x_3 \\
W &= 111,444
\end{align*}
\]
3-satisfiability reduces to subset sum

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

**Pf.** \( \implies \) Suppose \( \Phi \) is satisfiable.
- Choose integers corresponding to each \textit{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.

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

\( \neg x_1 \lor x_2 \lor x_3 \)

\( x_1 \lor \neg x_2 \lor x_3 \)

\( \neg x_1 \lor \neg x_2 \lor \neg x_3 \)

**My hobby**

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

\( \neg x_1 \lor x_2 \lor x_3 \)

\( x_1 \lor \neg x_2 \lor x_3 \)

\( \neg x_1 \lor \neg x_2 \lor \neg x_3 \)

\[
\begin{array}{cccccc}
  x_1 & x_2 & x_3 & C_1 & C_2 & C_3 \\
  0 & 0 & 0 & 1 & 0 & 100 \\
  0 & 0 & 0 & 2 & 0 & 200 \\
  0 & 0 & 0 & 0 & 1 & 10 \\
  0 & 0 & 0 & 0 & 0 & 20 \\
  0 & 0 & 0 & 0 & 0 & 1 \\
  0 & 0 & 0 & 0 & 2 & 1 \\
\end{array}
\]

\( \text{dummies to get clause} \)

\( \text{columns to sum to} 4 \)

\[
\begin{array}{cccccc}
  x_1 & x_2 & x_3 & C_1 & C_2 & C_3 \\
  0 & 0 & 0 & 1 & 0 & 100 \\
  0 & 0 & 0 & 2 & 0 & 200 \\
  0 & 0 & 0 & 0 & 1 & 10 \\
  0 & 0 & 0 & 0 & 0 & 20 \\
  0 & 0 & 0 & 0 & 0 & 1 \\
  0 & 0 & 0 & 0 & 2 & 1 \\
\end{array}
\]

\( \text{dummies to get clause} \)

\( \text{columns to sum to} 4 \)

**Partition**

\( \text{SUBSET-SUM.} \) 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 \( \frac{1}{2} \sum v_i \)?

**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.
  - \( v_1 = w_1, v_2 = w_2, \ldots, v_n = w_n, v_{n+1} = \sum w_j - W, v_{n+2} = \sum w_j + W \)
  - 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. \( \blacksquare \)
Scheduling with release times

**Theorem.** $\text{SUBSET-SUM} \leq_p \text{SCHEDULE}$.  

**Pf.** Given $\text{SUBSET-SUM}$ instance $w_1, \ldots, w_n$ and target $W$, construct an instance of $\text{SCHEDULE}$ that is feasible iff there exists a subset that sums to exactly $W$.

**Construction.**
- Create $n$ jobs with processing time $t_j = w_j$, release time $r_j = 0$, and no deadline ($d_j = 1 + \sum_j w_j$).
- Create job 0 with $t_0 = 1$, release time $r_0 = W$, and deadline $d_0 = W + 1$.
- Lemma: subset that sums to $W$ iff there exists a feasible schedule. \(\blacksquare\)

---

**Polynomial-time reductions**

<table>
<thead>
<tr>
<th>3-SAT</th>
<th>INDEPENDENT-SET</th>
<th>CLIQUE</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>DIR-HAM-CYCLE</td>
<td>NODE COVER</td>
</tr>
<tr>
<td></td>
<td>GRAPH-3-COLOR</td>
<td>SET PACKING</td>
</tr>
<tr>
<td></td>
<td>SUBSET-SUM</td>
<td>SATISFISABILITY WITH AT MOST 3 LITERAL PER CASE</td>
</tr>
<tr>
<td></td>
<td>VERTEX-COVER</td>
<td>SATISFICATION</td>
</tr>
<tr>
<td></td>
<td>HAM-CYCLE</td>
<td>0-1 INTEGER PROGRAMMING</td>
</tr>
<tr>
<td></td>
<td>PLANAR-3-COLOR</td>
<td>DIRECTED HAMILTON CIRCUIT</td>
</tr>
<tr>
<td></td>
<td>SCHEDULING</td>
<td>SET COVERING</td>
</tr>
<tr>
<td></td>
<td>TSP</td>
<td>UNDIRECTED HAMILTON CIRCUIT</td>
</tr>
<tr>
<td></td>
<td>SET-COVER</td>
<td>FEEDBACK NODE COVER</td>
</tr>
<tr>
<td></td>
<td></td>
<td>FEEDBACK ARC COVER</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SATISFICATION</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0-1 INTEGER</td>
</tr>
<tr>
<td></td>
<td></td>
<td>PROGRAMMING</td>
</tr>
<tr>
<td></td>
<td></td>
<td>DIRECTED</td>
</tr>
<tr>
<td></td>
<td></td>
<td>HAMILTON</td>
</tr>
<tr>
<td></td>
<td></td>
<td>CIRCUIT</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SET</td>
</tr>
<tr>
<td></td>
<td></td>
<td>COVERING</td>
</tr>
<tr>
<td></td>
<td></td>
<td>UNDIRECTED</td>
</tr>
<tr>
<td></td>
<td></td>
<td>HAMILTON</td>
</tr>
<tr>
<td></td>
<td></td>
<td>CIRCUIT</td>
</tr>
<tr>
<td></td>
<td></td>
<td>FEEDBACK</td>
</tr>
<tr>
<td></td>
<td></td>
<td>NODE</td>
</tr>
<tr>
<td></td>
<td></td>
<td>COVER</td>
</tr>
</tbody>
</table>

**Karp's 21 NP-complete problems**