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.

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 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.
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 $Y$.

computational model supplemented by special piece of hardware that solves instances of $Y$ in a single step
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 $Y \leq_p X$. 
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. $\text{FIND-MAX-FLOW} \leq_p \text{FIND-MIN-CUT}.$

B. $\text{FIND-MIN-CUT} \leq_p \text{FIND-MAX-FLOW}.$

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 $\leq 3$?
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.
Vertex cover and independent set reduce to one another

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

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

[Diagram showing a graph with black and white nodes, indicating an independent set of size 6 and a vertex cover of size 4]
Vertex cover and independent set reduce to one another

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

**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.
  \[ \Rightarrow \text{either } u \in V - S, \text{ or } v \in V - S, \text{ or both.} \]
- Thus, \( V - S \) covers \( (u, v) \).  

\[ \text{\blacksquare} \]
Vertex cover and independent set reduce to one another

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

**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 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. \( \blacksquare \)
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.

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

a set cover instance
Intractability: quiz 4

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
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 \( v \in V: S_v = \{ e \in E : e \text{ incident to } v \} \).

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

---

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

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

- $U = \{ 1, 2, 3, 4, 5, 6, 7 \}$
- $S_a = \{ 3, 7 \}$
- $S_b = \{ 2, 4 \}$
- $S_c = \{ 3, 4, 5, 6 \}$
- $S_d = \{ 5 \}$
- $S_e = \{ 1 \}$
- $S_f = \{ 1, 2, 6, 7 \}$
**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 $(U, S, k)$.
- Then $X = \{ v : S_v \in Y \}$ is a vertex cover of size $k$ in $G$.  

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

- “no” instances of VERTEX-COVER are solved correctly

- vertex cover instance (k = 2)
- set cover instance (k = 2)
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. \( C_j = x_1 \lor \overline{x_2} \lor x_3 \)

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).

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

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 exist a poly-time algorithm for 3-SAT.

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

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

Lemma.  Φ is satisfiable iff $G$ contains an independent set of size $k = |Φ|$.

Pf.  ⇒  Consider any satisfying assignment for Φ.

• Select one true literal from each clause/triangle.
• This is an independent set of size $k = |Φ|$.

"yes" instances of 3-SAT are solved correctly

$k = 3$

$$\Phi = (\overline{x_1} \lor x_2 \lor x_3) \land (x_1 \lor \overline{x_2} \lor x_3) \land (\overline{x_1} \lor x_2 \lor x_4)$$
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. \( \blacksquare \)

\[
G
\]

\[
k = 3
\]

\[
\Phi = (\overline{x_1} \lor x_2 \lor x_3) \land (x_1 \lor \overline{x_2} \lor x_3) \land (\overline{x_1} \lor x_2 \lor x_4)
\]
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. $3\text{-SAT} \leq_p \text{INDEPENDENT-SET} \leq_p \text{VERTEX-COVER} \leq_p \text{SET-COVER}$.
**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.
**Search Problems vs. Decision Problems**

**Vertex-Cover.** Does there exist a vertex cover of size \( \leq k \)?

**Find-Vertex-Cover.** Find a vertex cover of size \( \leq k \).

**Theorem.** \( \text{Vertex-Cover} \equiv_p \text{Find-Vertex-Cover} \).

**Pf. \( \leq_p \)** Decision problem is a special case of search problem. □

**Pf. \( \geq_p \)**

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 \( G - \{ v \} \). □

delete \( v \) and all incident edges
**Optimization Problems vs. Search Problems**

**Find-Vertex-Cover.** Find a vertex cover of size \( \leq k \).

**Find-Min-Vertex-Cover.** Find a vertex cover of minimum size.

**Theorem.** \( \text{Find-Vertex-Cover} \equiv_p \text{Find-Min-Vertex-Cover} \).

**Pf.** \( \leq_p \) Search problem is a special case of optimization problem.

**Pf.** \( \geq_p \) To find vertex cover of minimum size:

- Binary search (or linear search) for size \( k^* \) of min vertex cover.
- Solve search problem for given \( k^* \).
8. **Intractability I**

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

Section 8.5
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?

![Diagram of a graph with vertices 1, 2, 3, 4, 5 and their corresponding primed vertices, showing no Hamilton cycle exists.]
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.** $\text{DIRECTED-HAMILTON-CYCLE} \leq_p \text{HAMILTON-CYCLE}.$

**Pf.** Given a directed graph $G = (V, E)$, construct a graph $G'$ with $3n$ nodes.
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:
  - $\ldots, \text{black}, \text{white}, \text{blue}, \text{black}, \text{white}, \text{blue}, \text{black}, \text{white}, \text{blue}, \ldots$
  - $\ldots, \text{black}, \text{blue}, \text{white}, \text{black}, \text{blue}, \text{white}, \text{black}, \text{blue}, \text{white}, \ldots$
- 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-SAT $\leq_p$ DIRECTED-HAMILTON-CYCLE.

**Pf.** Given an instance $\Phi$ of 3-SAT, we construct an instance $G$ of 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 \( \iff \) 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.

```
x_i = true  \iff \text{node for clause } C_j \text{ is connected in this way if } x_i \text{ appears in clause } C_j
```

```
x_i = false \iff \text{node for clause } C_k \text{ is connected in this way if } \overline{x_i} \text{ appears in clause } C_k
```
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

**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)  •
3-satisfiability reduces to directed Hamilton cycle

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

Pf. $\Leftarrow$

• 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. □
Poly-time reductions

constraint satisfaction

3-Sat

INDEPENDENT-SET

DIR-HAM-CYCLE

3-COLOR

SUBSET-SUM

VERTEX-COVER

HAM-CYCLE

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
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>Tardos</td>
<td>COS 523</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?

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

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

\[
T_4 = \{ x_2, y_2, z_3 \}, \quad T_5 = \{ x_2, y_3, z_3 \},
\]

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

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

**Remark.** Generalization of bipartite matching.
3-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 $\leq_p$ 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.
Construction. (part 1)

• Create gadget for each variable $x_i$ with $2k$ core elements and $2k$ tip ones.

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 \( x_i = true \)) or all blue ones (corresponding to \( x_i = false \)).

![Diagram](image-url)
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 $x_3$.

$$C_1 = x_1 \lor \overline{x_2} \lor x_3$$
3-satisfiability reduces to 3-dimensional matching

**Construction.** (part 3)
- There are $2nk$ tips: $nk$ covered by blue/gray 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.

\[
C_1 = x_1 \lor \overline{x_2} \lor x_3
\]
3-satisfiability reduces to 3-dimensional matching

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

**Q.** What are \(X, Y,\) and \(Z\)?

\[
C_1 = x_1 \lor \overline{x_2} \lor x_3
\]
3-satisfiability reduces to 3-dimensional matching

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

**Q.** What are \(X\), \(Y\), and \(Z\)?

**A.** \(X = \text{black},\ Y = \text{white},\ \text{and}\ Z = \text{blue}\).

\[
C_1 = x_1 \lor \overline{x_2} \lor x_3
\]
3-satisfiability reduces to 3-dimensional matching

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

**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. 

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

---

clause 1 gadget

\(x_1\)

\(x_2\)

\(x_3\)

true

false

clause 1 tips

core

cleanup gadget

...
8. Intractability I

- poly-time reductions
- packing and covering problems
- constraint satisfaction problems
- sequencing problems
- partitioning problems
- graph coloring
- numerical problems
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?
Intractability: quiz 6

How difficult to solve 2-COLOR?

A. $O(m + n)$ using BFS or DFS.

B. $O(mn)$ using maximum flow.

C. $\Omega(2^n)$ using brute force.

D. Not even Tarjan knows.
Application: register allocation

Register allocation. Assign program variables to machine registers 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; 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.** $\text{3-Sat} \leq_p \text{3-Color}$. 

**Pf.** Given $\text{3-Sat}$ instance $\Phi$, we construct an instance of $\text{3-Color}$ that is 3-colorable iff $\Phi$ is satisfiable.
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.

\[ \text{to be described later} \]
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).
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*.

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

$3$-satisfiability reduces to $3$-colorability
3-satisfiability reduces to 3-colorability

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

**Pf.** $\iff$ 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. □

\[ a \text{ literal set to true in 3-SAT assignment} \]

\[ C_j = x_1 \lor \overline{x_2} \lor x_3 \]
Poly-time reductions

constraint satisfaction

3-Sat

- INDEPENDENT-SET
  - VERTX-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 $W$?


**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.
**Subset sum**

**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.

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

\[
\begin{array}{llllll}
& x_1 & x_2 & x_3 & C_1 & C_2 & C_3 \\
\hline
x_1 & 1 & 0 & 0 & 0 & 1 & 0 \\
\neg x_1 & 1 & 0 & 0 & 1 & 0 & 1 \\
x_2 & 0 & 1 & 0 & 1 & 0 & 0 \\
\neg x_2 & 0 & 1 & 0 & 0 & 1 & 1 \\
x_3 & 0 & 0 & 1 & 1 & 1 & 0 \\
\neg x_3 & 0 & 0 & 1 & 0 & 0 & 1 \\
\hline
& 0 & 0 & 0 & 1 & 0 & 0 \\
& 0 & 0 & 0 & 2 & 0 & 0 \\
& 0 & 0 & 0 & 0 & 1 & 0 \\
& 0 & 0 & 0 & 0 & 2 & 0 \\
& 0 & 0 & 0 & 0 & 0 & 1 \\
& 0 & 0 & 0 & 0 & 0 & 2 \\
\hline
W & 1 & 1 & 1 & 4 & 4 & 4
\end{array}
\]

3-SAT instance

subset-sum 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 \( x_i^* = true \), 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.

\[ C_1 = \neg x_1 \vee x_2 \vee x_3 \]

\[ C_2 = x_1 \vee \neg x_2 \vee x_3 \]

\[ C_3 = \neg x_1 \vee \neg x_2 \vee \neg x_3 \]

\[
\begin{array}{cccccc}
\text{W} & 1 & 1 & 1 & 4 & 4 & 4 & 111,444 \\
\hline
x_1 & 1 & 0 & 0 & 0 & 1 & 0 & 100,010 \\
\neg x_1 & 1 & 0 & 0 & 1 & 0 & 1 & 100,101 \\
x_2 & 0 & 1 & 0 & 1 & 0 & 0 & 10,100 \\
\neg x_2 & 0 & 1 & 0 & 0 & 1 & 1 & 10,011 \\
x_3 & 0 & 0 & 1 & 1 & 1 & 0 & 1,110 \\
\neg x_3 & 0 & 0 & 1 & 0 & 0 & 1 & 1,001 \\
\end{array}
\]
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 $x_i^* = \text{true}$; otherwise, assign $x_i^* = \text{false}$.

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

\[
\begin{align*}
\text{3-SAT instance} & \\
C_1 &= \neg x_1 \lor x_2 \lor x_3 \\
C_2 &= x_1 \lor \neg x_2 \lor x_3 \\
C_3 &= \neg x_1 \lor \neg x_2 \lor \neg x_3
\end{align*}
\]

\[
\begin{array}{cccccc}
\text{subset-sum instance} \\
\hline
\text{row} & x_1 & x_2 & x_3 & C_1 & C_2 & C_3 \\
\hline
x_1 & 1 & 0 & 0 & 0 & 1 & 0 \\
\neg x_1 & 1 & 0 & 0 & 1 & 0 & 1 \\
x_2 & 0 & 1 & 0 & 1 & 0 & 0 \\
\neg x_2 & 0 & 1 & 0 & 0 & 1 & 1 \\
x_3 & 0 & 0 & 1 & 1 & 1 & 0 \\
\neg x_3 & 0 & 0 & 1 & 0 & 0 & 1 \\
\hline
0 & 0 & 0 & 1 & 0 & 0 & 100 \\
0 & 0 & 0 & 2 & 0 & 0 & 200 \\
0 & 0 & 0 & 0 & 1 & 0 & 10 \\
0 & 0 & 0 & 0 & 2 & 0 & 20 \\
0 & 0 & 0 & 0 & 0 & 1 & 1 \\
0 & 0 & 0 & 0 & 0 & 2 & 2 \\
\hline
W & 1 & 1 & 1 & 4 & 4 & 4 & 111,444
\end{array}
\]
**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

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

- Packing and covering
- Sequencing
- Partitioning
- Numerical
Karp’s 20 poly-time reductions from satisfiability

![Diagram of reductions from satisfiability to other problems]

**Dick Karp (1972)**

1985 Turing Award

**Figure 1** - Complete Problems