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. \( O(n^4) \) algorithm unlikely.
• PSPACE-completeness. \( O(n^4) \) 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>

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 $Y \leq_P X$.  

Frustrating news. Huge number of fundamental problems have defied classification for decades.
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 $X$.

Intractability: quiz 2

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 $X \not\leq_p Y$ 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 $\leq 3$?

**Intractability: quiz 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.** INDEPENDENT-SET $\equiv_p$ VERTEX-COVER.

**Pf.** We show $S$ is an independent set of size $k$ iff $V - S$ is a vertex cover of size $n - k$. 
Vertex cover and independent set reduce to one another

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

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

\begin{itemize}
  \item Let \( S \) be any independent set of size \( k \).
  \item \( V - S \) is of size \( n - k \).
  \item Consider an arbitrary edge \( (u,v) \in E \).
  \item \( S \) independent \( \Rightarrow \) either \( u \notin S \), or \( v \notin S \), or both.
  \item \( \Rightarrow \) either \( u \in V - S \), or \( v \in V - S \), or both.
  \item Thus, \( V - S \) covers \((u,v)\). \n\end{itemize}

Set cover

\textbf{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.

\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}

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

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?

\begin{itemize}
  \item \textbf{A.} 1
  \item \textbf{B.} 2
  \item \textbf{C.} 3
  \item \textbf{D.} None of the above.
\end{itemize}

\[
U = \{1, 2, 3, 4, 5, 6, 7\} \\
S_a = \{1, 4, 6\} \\
S_b = \{1, 6, 7\} \\
S_c = \{1, 2, 3, 6\} \\
S_d = \{1, 3, 5, 7\} \\
S_e = \{2, 6, 7\} \\
S_f = \{3, 4, 5\}
\]
Lemma. \( G = (V, E) \) contains a vertex cover of size \( k \) iff \((U, S, k)\) contains a set cover of size \( k \).

Construction.
- Universe \( U = \{ e \} \) and \( \{ s \} \), we construct a vertex cover instance \((U, S, k)\) that has a set cover of size \( k \).

Proof. Let \( X \subseteq V \) be a vertex cover of size \( k \).
- Then \( Y = \{ s : v \in X \} \) is a set cover of size \( k \).

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

Proof. Let \( Y \subseteq S \) be a set cover of size \( k \).
- Then \( X = \{ v : s \in Y \} \) is a vertex cover of size \( k \).
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).

3-satisfiability reduces to independent set

Theorem. \( 3\text{-SAT} \leq_p \text{INDEPENDENT-SET} \).

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.

Construction.

\( \bullet \) \( G \) contains 3 nodes for each clause, one for each literal.
\( \bullet \) \( G \) connects 3 literals in a clause in a triangle.
\( \bullet \) \( G \) connects literal to each of its negations.

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 is hard

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

P vs. NP. This hypothesis is equivalent to \( \mathbf{P} \neq \mathbf{NP} \) conjecture.

Satisfiability is hard

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

P vs. NP. This hypothesis is equivalent to \( \mathbf{P} \neq \mathbf{NP} \) conjecture.

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 \( \Phi \).

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

"yes" instances of 3-SAT are solved correctly

https://www.facebook.com/pg/npcompleteteens

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 \( \Phi \).

\( \bullet \) Select one true literal from each clause/triangle.
\( \bullet \) This is an independent set of size \( k = |\Phi| \). □
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 \( \text{true} \) (and remaining literals consistently).
- All clauses in \( \Phi \) are satisfied.

> "no" instances of 3-SAT are solved correctly

\[
\begin{align*}
\Phi = & \left( x_1 \lor x_2 \lor x_3 \right) \\
& \land \left( x_1 \lor \overline{x_2} \lor x_5 \right) \\
& \land \left( \overline{x_1} \lor x_2 \lor x_4 \right)
\end{align*}
\]

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 \( \leq k - 1 \).
- 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.** \textsc{Find-Vertex-Cover} \( \equiv_p \) \textsc{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 \( k^* \). •

**Hamilton Cycle**

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

---

**8. Intractability I**

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

**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.

\[ \text{directed graph } G \]

\[ \text{undirected graph } G' \]

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 \( x_i = \text{true} \).

\[ \text{directed graph } G \]
**Intractability: quiz 5**

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.

---

**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 $G - \{C_1, C_2, ..., C_k\}$.
- Set $x_i^j = true$ if $\Gamma'$ traverses row $i$ left-to-right; otherwise, set $x_i^j = false$.
- traversed in "correct" direction, and each clause is satisfied. 

---

### 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, $m$ 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?

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

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

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

\[ T_6 = \{ x_3, y_1, z_3 \}, \quad T_7 = \{ x_3, y_1, z_2 \} \]

\[ T_8 = \{ x_3, y_2, z_1 \} \]

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.

**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 $x_i = \text{false}$).
3-satisfiability reduces to 3-dimensional matching

**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 \(x_3\).

\[
\text{clause 1 gadget } \quad C_1 = x_1 \lor \overline{x_2} \lor x_3
\]

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

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. $3$-COLOR $\leq_p$ K-REGISTER-ALLOCATION for any constant $k \geq 3$.

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 $C_i$, add a gadget of 6 nodes and 13 edges.

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.

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

6-node gadget

**Pf.** $\Leftarrow$ 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.

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

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.

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

Poly-time reductions

**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.

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

suppose, for the sake of contradiction, that all 3 literals are white in some 3-coloring
contradiction (not a 3-coloring)

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

3-SAT

constraint satisfaction

packing and covering
sequencing
partitioning
numerical


8. **INTRACTABILITY**

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

**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.** $3$-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 having $n + k$ digits:
- Include one digit for each variable $x_i$ and one digit for each clause $C_j$.
- Include two digits 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.

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

Pf. $\Rightarrow$ 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.

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

Pf. $\Rightarrow$ 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.

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 $w_i \geq 0$, values $v_i \geq 0$, a weight limit $U$, and a target value $V$, is there a subset $S \subseteq X$ such that:

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

Recall. $O(nU)$ 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

Karp's 20 poly-time reductions from satisfiability

Dick Karp (1972)
1985 Turing Award