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. $O(n^r)$ 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.


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

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>

Poly-time reductions

Desiderata. Suppose we could solve problem \( Y \) in polynomial time. What else could we solve in polynomial time?

Reduction. Problem \( X \) can be solved using:
- Polynomial number of standard computational steps, plus
- Polynomial number of calls to oracle that solves problem \( Y \).

\( X \leq_p Y \)

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.

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?

Poly-time reductions

Desiderata. Suppose we could solve problem \( Y \) in polynomial time. What else could we solve in polynomial time?

Reduction. Problem \( X \) can be solved using:
- Polynomial number of standard computational steps, plus
- Polynomial number of calls to oracle that solves problem \( Y \).

Notation. \( X \leq_p Y \).

\( X \leq_p Y \) with \( Y \leq_p X \).
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. $\text{FIND-MAX-FLOW} \leq_p \text{FIND-MIN-CUT}$.
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 $X \not\leq_p Y$ 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.
**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.** \( \text{INDEPENDENT-SET} \equiv_p \text{VERTEX-COVER} \).

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

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

\[
\begin{align*}
\text{\( \Leftarrow \)} & \quad \text{Let } V - S \text{ be any vertex cover of size } n - k.
\text{\( \Leftarrow \)} & \quad S \text{ is of size } k.
\text{\( \Leftarrow \)} & \quad \text{Consider an arbitrary edge } (u, v) \in E.
\text{\( \Leftarrow \)} & \quad V - S \text{ is a vertex cover } \Rightarrow \text{ either } u \notin V - S \text{, or } v \notin V - S \text{, or both.}
\text{\( \Leftarrow \)} & \quad \text{either } u \notin S \text{, or } v \notin S \text{, or both.}
\text{\( \Leftarrow \)} & \quad \text{Thus, } S \text{ is an independent set.}
\end{align*}
\]

---

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.**

\[
\begin{align*}
\text{\textbullet } & \quad m \text{ available pieces of software.}
\text{\textbullet } & \quad \text{Set } U \text{ of } n \text{ capabilities that we would like our system to have.}
\text{\textbullet } & \quad \text{The } i^{th} \text{ piece of software provides the set } S_i \subseteq U \text{ of capabilities.}
\text{\textbullet } & \quad \text{Goal: achieve all } n \text{ capabilities using fewest pieces of software.}
\end{align*}
\]

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

\begin{tikzpicture}
\node (a) at (0,0) {a set cover instance};
\end{tikzpicture}

---

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?

\[
\begin{align*}
\text{A. } & \quad 1 \\
\text{B. } & \quad 2 \\
\text{C. } & \quad 3 \\
\text{D. } & \quad \text{None of the above.}
\end{align*}
\]

\[
\begin{align*}
U & = \{ 1, 2, 3, 4, 5, 6, 7 \} \\
S_a & = \{ 1, 4, 6 \} \quad S_b = \{ 1, 6, 7 \} \\
S_c & = \{ 1, 2, 3, 6 \} \quad S_d = \{ 1, 3, 5, 7 \} \\
S_e & = \{ 2, 6, 7 \} \quad S_f = \{ 3, 4, 5 \}
\end{align*}
\]
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 \( v \in V : S_v = \{ e \in E : e \text{ incident to } v \} \).

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

Vertex cover 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.**
- 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 \( k \).

---

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.

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

Satisfiability

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

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

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

Construction.

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

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

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 \( k = |\Phi| \).

"yes" instances of 3-SAT are solved correctly
3-satisfiability reduces to independent set

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

**Pf.** ⇔ Let $S$ be independent set of size $k$.
- $S$ must contain exactly one node in each triangle.
- Set these literals to $true$ (and remaining literals consistently).
- All clauses in $\Phi$ are satisfied. ♦

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

**Search problems vs. decision problems**

**VERTEX-COVER.** Does there exist a vertex cover of size $\le k$?

**Find-VERTEX-COVER.** Find a vertex cover of size $\le k$.

**Theorem.** $\text{VERTEX-COVER} \le_p \text{Find-VERTEX-COVER}$.

**Pf.** $\le_p$ Decision problem is a special case of search problem. ♦

**Pf.** $\ge_p$

To find a vertex cover of size $\le k$:
- Determine if there exists a vertex cover of size $\le k$.
- Find a vertex $v$ such that $G - \{v\}$ has a vertex cover of size $\le k - 1$.
- Any vertex in any vertex cover of size $\le k$ will have this property.
- Include $v$ in the vertex cover.
- Recursively find a vertex cover of size $\le k - 1$ in $G - \{v\}$. ♦

**Review**

**Basic reduction strategies.**
- Simple equivalence: $\text{INDEPENDENT-SET} =_p \text{VERTEX-COVER}$.
- Special case to general case: $\text{VERTEX-COVER} \le_p \text{SET-COVER}$.
- Encoding with gadgets: $3\text{-SAT} \le_p \text{INDEPENDENT-SET}$.

**Transitivity.** If $X \le_p Y$ and $Y \le_p Z$, then $X \le_p Z$.

**Pf idea.** Compose the two algorithms.

**Ex.** $3\text{-SAT} \le_p \text{INDEPENDENT-SET} \le_p \text{VERTEX-COVER} \le_p \text{SET-COVER}$.  

**Decision, search, and optimization problems**

**Decision problem.** Does there exist a vertex cover of size $\le k$?

**Search problem.** Find a vertex cover of size $\le k$.

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

**Goal.** Show that all three problems poly-time reduce to one another.
**Finding 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.** Find-Vertex-Cover \( \leq_p \) 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?
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.

\[ \text{directed graph } G \]

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

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 $x_i = true.$
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.

\[ C_j = x_i \lor x_j \lor x_k \]

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 \( C_i \) 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 $C_j$, 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 $x_i^j = true$ if $\Gamma'$ traverses row $i$ left-to-right; otherwise, set $x_i^j = false$.
- traversed in “correct” direction, and each clause is satisfied.  

Poly-time reductions

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-satisfiability reduces to 3-dimensional matching

**Construction.** (part 1)

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

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 $x_i = false$).
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 $x_3$.

3-satisfiability reduces to 3-dimensional matching

**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.

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.** $X = black$, $Y = white$, and $Z = blue$. 

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.** $X = black$, $Y = white$, and $Z = blue$. 

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 $x_3$.

3-satisfiability reduces to 3-dimensional matching

**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.

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.** $X = black$, $Y = white$, and $Z = blue$. 

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.** $X = black$, $Y = white$, and $Z = blue$. 

3-satisfiability reduces to 3-dimensional matching

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

\[
Pf. \quad \Rightarrow \quad \text{If 3d-matching, then assign } x_i \text{ according to gadget } x_i.

\[
Pf. \quad \Leftarrow \quad \text{If } \Phi \text{ is satisfiable, use any true literal in } C_j \text{ to select gadget } C_j \text{ triple.}
\]

8. Intractability

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

---

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

*Construction.*
1. Create a graph $G$ with a node for each literal.
2. Connect each literal to its negation.
3. Create 3 new nodes $T$, $F$, and $B$; connect them in a triangle.
4. Connect each clause $C_i$ to a gadget of 6 nodes and 13 edges.
5. To be described later

---

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

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

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

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 \textit{true} literals \textit{black} and all \textit{false} literals \textit{white}.
- Pick one \textit{true} literal; color node below that node \textit{white}, and node below that \textit{blue}.
- Color remaining middle row nodes \textit{blue}.
- Color remaining bottom nodes \textit{black} or \textit{white}, as forced.

Poly-time reductions
### Subset sum

**Subet-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.** \( \{ 215, 215, 275, 275, 355, 355, 420, 420, 580, 580, 655, 655 \} \), \( W = 1505 \).

**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.

---

### Subset sum

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

**Pf.** Given an instance \( \Phi \) of 3-SAT, we construct an instance of Subset-Sum that has a 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 \( \neg x_i \) digit is 4.

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

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

\[\begin{array}{c}
\text{dummies to get clause columns to sum to 4} \\
\hline
\begin{array}{cccccccc}
\neg x_1 & x_1 & \neg x_2 & x_3 & x_1 & \neg x_2 & \neg x_3 \\
0 & 1 & 0 & 0 & 1 & 0 & 0 & 1
\end{array}
\end{array}\]

\[\begin{array}{cccccccc}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 2
\end{array}\]

**SUBSET-SUM instance**

\[
\begin{array}{cccccccc}
W & 1 & 1 & 1 & 4 & 4 & 4 & 4
\end{array}
\]

\[
\begin{array}{cccccccc}
111,444
\end{array}
\]

### 3-satisfiability reduces to subset sum

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

**Pf.** \( \Leftarrow \) 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. 

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

\[\begin{array}{c}
\text{dummies to get clause columns to sum to 4} \\
\hline
\begin{array}{cccccccc}
\neg x_1 & x_1 & \neg x_2 & x_3 & x_1 & \neg x_2 & \neg x_3 \\
0 & 1 & 0 & 0 & 1 & 0 & 1
\end{array}
\end{array}\]

\[\begin{array}{cccccccc}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 2
\end{array}\]

**SUBSET-SUM instance**

\[
\begin{array}{cccccccc}
W & 1 & 1 & 1 & 4 & 4 & 4 & 4
\end{array}
\]

\[
\begin{array}{cccccccc}
111,444
\end{array}
\]

### 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 \( w_i \geq 0 \), 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.** \( O(n U) \) dynamic programming algorithm for KNAPSACK.

**Challenge.** Prove \( \text{SUBSET-SUM} \trianglelefteq \text{KNAPSACK} \).

**Pf.** Given instance \( (w_1, \ldots, w_n, W) \) of \( \text{SUBSET-SUM} \), create \( \text{KNAPSACK} \) instance:
Poly-time reductions

3-SAT

constraint satisfaction

INDEPENDENT-SET

DIR-HAM-CYCLE

3-COLOR

SUBSET-SUM

VERTEX-COVER

HAM-CYCLE

KNAPSACK

packing and covering

sequencing

partitioning

numerical

Karp’s 20 poly-time reductions from satisfiability

Dick Karp (1972)

1985 Turing Award

Dick Karp (1972)

1985 Turing Award