NP-Complete Problems So Far

**Theorem:** INDEPENDENT-SET, VERTEX-COVER, SET-COVER, SAT, 3-SAT, HAM-CYCLE, HAM-PATH, TSP are all NP-Complete.

Arrows show reductions discussed in class.
We could construct a polynomial reduction between any pair.

Numerical problems

**Subset Sum** decision problem: given \( n \) items with weights \( w_1, \ldots, w_n \), is there a subset of items whose weight is exactly \( W \)?

Dynamic programming: \( O(nW) \) pseudo-polynomial time algorithm
(not polynomial in input length \( n \log W \))

Subset Sum (cont.)

- Set \( n + j \)th digit of \( W = 3 \)
- Consider a subset of items corresponding to a truth assignment (exactly one of \( t_i, f_i \))
  - If \( C_j \) is not satisfied, then total in position \( n + j \) is 0, otherwise it is 1, 2, or 3
  - Create two “dummy” items \( y_j, z_j \) with 1 in position \( n + j \)
  - Can select dummies to yield total of 3 in position \( n + j \) iff \( C_j \) is satisfied
### Subset Sum Example

**Example.**

\[(x_1 \lor \neg x_2 \lor x_3) \land (\neg x_1 \lor x_2 \lor \neg x_3) \land (\neg x_1 \lor \neg x_2 \lor x_3)\]

<table>
<thead>
<tr>
<th>Item</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>(t_1)</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>(f_1)</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>(t_2)</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>(f_2)</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>(t_3)</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>(f_3)</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Item</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>(y_1)</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>(z_1)</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>(y_2)</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>(z_2)</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>(y_3)</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>(z_3)</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

\(W\) | 1 | 1 | 1 | 3 | 3 | 3

### Warning

**Theorem.** **SubsetSum** is NP-Complete.

But Subset Sum can be tricky!

- If reducing **SubsetSum** \(\leq_p X\), reduction needs to be polynomial in \(\log(W)\) (number of digits).

### Reduction

- Reduce from 3-SAT.

Skeleton: 1 color for True, 1 for False

3 extra nodes in a clique \(T, F, B\).

For each variable \(x_i\), two nodes \(v_{0i}, v_{1i}\).

Edges \((v_{0i}, B), (v_{1i}, B), (v_{0i}, v_{1i})\).

Either \(v_{0i}\) or \(v_{1i}\) gets the \(T\) color.

### Graph Coloring

**Def.** A \(k\)-coloring of a graph \(G = (V, E)\) is a function \(f : V \rightarrow \{1, \ldots, k\}\) such that for all \((u, v) \in E\), \(f(u) \neq f(v)\).

**Problem.** Given \(G = (V, E)\) and number \(k\), does \(G\) have a \(k\)-coloring?

Many applications

- Actually coloring maps!
- Scheduling jobs on machine with competing resources.
- Allocating variables to registers in a compiler.

**Claim.** 2-coloring \(\in P\) (equivalent to bipartite testing)

**Theorem.** 3-coloring is NP-Complete.
Proof

- Graph is polynomial in $n + m$.
- If satisfying assignment
  - Color $B, T, F$ then $v_{11}$ as $T$ if $\phi(x_i) = 1$.
  - Since clauses satisfied, can color each gadget.
- If graph 3-colorable
  - One of $v_{10}, v_{11}$ must get $T$ color.
  - Clause gadget colorable iff clause satisfied.

Question. What about $k$-coloring?

Clicker Question 2

Which of the following is true?

A: If we can reduce 3-coloring to $k$-coloring, then $k$-coloring is NP-complete
B: $k$-coloring is NP-complete since any 3-coloring is also a $k$-coloring for $k \geq 3$
C: $k$-coloring is not NP-complete since 3-coloring is the hardest case, for $k > 3$ the coloring is easier
D: $k$-coloring is not NP-complete because the 4-color theorem has been proved

NP-Completeness Recap

Types of hard problems:

... any many others. See book or other sources for more examples.
You can use any known NP-complete problem to prove a new problem is NP-complete.