NP-Completeness and Reductions

Careful, direction of reduction matters!

A \leq_P B: A reduces to B (A "no harder" than B) From arbitrary instance of A, construct instance of B
Reduction and construction is one-way

Problem instances are equivalent (both ways): \text{YES}_A \implies \text{YES}_B
\text{YES}_B \implies \text{YES}_A \text{ (same as } \text{NO}_A \implies \text{NO}_B\text{)}

B is NP-complete means:
1. B is in NP: can check solution in polynomial time ("easy enough")
2. B is NP-hard: some NP-complete A reduces to B: A \leq_P B ("hard enough")
   we also say: reduce from A

Clicker Question

Which of the following graph problems are in NP?

A. Length of longest simple path is \leq k
B. Length of longest simple path is = k
C. Length of longest simple path is \geq k
D. Find length of longest simple path.
E. All of the above.

Subset Sum Warmup

Does this instance have a solution?

\begin{tabular}{l l}
  w1 & 1010 \\
  w2 & 1001 \\
  w3 & 0110 \\
  w4 & 0101 \\
  ---- & ---- \\
  W & 1111 \\
\end{tabular}

Dynamic programming: \text{O}(nW) pseudo-polynomial time algorithm (not polynomial in input length n \log W)
Subset Sum Warmup

For which nonzero values of $y$ does this instance have a solution?

\[
\begin{array}{cccc}
10010 \\
10001 \\
01001 \\
01010 \\
00111 \\
00100 \\
\hline
1113y
\end{array}
\]

A. $y = 1$
B. $y = 1, 2$
C. $y = 1, 2, 3$

\[
\begin{array}{cccc}
10010 \\
10001 \\
01001 \\
01010 \\
00111 \\
00100 \\
\hline
1112y
\end{array}
\]

A. $y = 1$
B. $y = 1, 2$
C. $y = 1, 2, 3$

\[
\begin{array}{cccc}
10010 \\
10001 \\
01001 \\
01010 \\
00111 \\
00100 \\
\hline
1111y
\end{array}
\]

A. $y = 1$
B. $y = 1, 2$
C. $y = 1, 2, 3$

Subset Sum

Theorem. Subset sum is NP-complete.

Reduction from 3-SAT. ($n$ variables, $m$ clauses, base 10).

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

<table>
<thead>
<tr>
<th>Item</th>
<th>variable digits</th>
<th>clause digits</th>
</tr>
</thead>
<tbody>
<tr>
<td>$t_1$</td>
<td>1 0 0</td>
<td>1</td>
</tr>
<tr>
<td>$f_1$</td>
<td>1 0 0</td>
<td>1 0</td>
</tr>
<tr>
<td>$t_2$</td>
<td>0 1 0</td>
<td>1 0 1</td>
</tr>
<tr>
<td>$f_2$</td>
<td>0 1 0</td>
<td>1 0 1</td>
</tr>
<tr>
<td>$t_3$</td>
<td>0 0 1</td>
<td>1 0 1</td>
</tr>
<tr>
<td>$f_3$</td>
<td>0 0 1</td>
<td>1 0 1</td>
</tr>
<tr>
<td>$W$</td>
<td>1 1 1</td>
<td></td>
</tr>
</tbody>
</table>

- Items $t_i, f_i$ for each $x_i$; correspond to truth assignment
- Weights $\implies$ select exactly one
- (Numbers are base 10)

Subset Sum Reduction

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

<table>
<thead>
<tr>
<th>Item</th>
<th>variable digits</th>
<th>clause digits</th>
</tr>
</thead>
<tbody>
<tr>
<td>$t_1$</td>
<td>1 0 0</td>
<td>1 0 0</td>
</tr>
<tr>
<td>$f_1$</td>
<td>1 0 0</td>
<td>0 1 1</td>
</tr>
<tr>
<td>$t_2$</td>
<td>0 1 0</td>
<td>0 1 0</td>
</tr>
<tr>
<td>$f_2$</td>
<td>0 1 0</td>
<td>0 1 0</td>
</tr>
<tr>
<td>$t_3$</td>
<td>0 0 1</td>
<td>0 1 1</td>
</tr>
<tr>
<td>$f_3$</td>
<td>0 0 1</td>
<td>0 1 1</td>
</tr>
<tr>
<td>$W$</td>
<td>1 1 1</td>
<td>1</td>
</tr>
</tbody>
</table>

- Clause digit equal to 1 if $x_i$ assignment satisfies $C_j$
- Total for clause digit $> 0$ if assignment satisfies $C_j$
- Goal: all clause digits $> 0$. How to set $W$ to enforce this?
  - Total could be 1, 2, 3 for satisfied clause.
Theorem. \textsc{SubsetSum} is NP-Complete.

\begin{itemize}
  \item But reducing \textsc{from Subset Sum} can be tricky!
  \end{itemize}

\begin{itemize}
  \item If reducing \textsc{SubsetSum} \leq_P X, reduction needs to be polynomial in \log(W) (number of digits).
  \end{itemize}

\textbf{Warning}

For variable \( x_i \), create two items \( t_i, f_i \)
\begin{itemize}
  \item Both have \( i \)th digit equal to 1
  \item All other items have zero in this digit
  \item \( i \)th digit of \( W = 1 \) \( \Rightarrow \) select exactly one of \( t_i, f_i \)
\end{itemize}

The \( n + j \)th digit corresponds to clause \( C_j \)
\begin{itemize}
  \item If \( x_i \in C_j \), set \( n + j \)th digit of \( t_i = 1 \)
  \item If \( \neg x_i \in C_j \), set \( n + j \)th digit of \( f_i = 1 \)
  \item Everything else 0.
\end{itemize}

\begin{itemize}
  \item Set \( n + j \)th digit of \( W = 3 \)
  \item Consider a subset of items corresponding to a truth assignment (exactly one of \( t_i, f_i \))
  \item If \( C_j \) is not satisfied, then total in position \( n + j \) is 0, otherwise it is 1, 2, or 3
  \item Create two "dummy" items \( y_j, z_j \) with 1 in position \( n + j \)
  \item Can select dummies to yield total of 3 in position \( n + j \) iff \( C_j \) is satisfied
\end{itemize}
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.

---

Reduction: Clause Gadget

For clause $x_i \lor \neg x_j \lor x_k$

![Reduction Diagram](image)

Top node can be colored iff not all three $v$-nodes are $F$.

---

3-Color: Gadget for Variables

- Reduce from $3$-SAT.

3 colors: True, False, “Base”

3 special nodes in a clique $T, F, B$.

For each variable $x_i$, two nodes $v_{i0}, v_{i1}$.

Edges $(v_{i0}, B), (v_{i1}, T), (v_{i0}, v_{i1})$.

Either $v_{i0}$ or $v_{i1}$ colored $T$, the other colored $F$.

---

Proof

- Graph is polynomial in $n + m$.
- If satisfying assignment
  - Color $B, T, F$ then $v_{i1}$ as $T$ if $\phi(x_i) = 1$.
  - Since clauses satisfied, can color each gadget.
- If graph 3-colorable
  - One of $v_{i0}, v_{i1}$ must get $T$ color.
  - Clause gadget colorable iff clause satisfied.

**Question.** What about $k$-coloring?

---

Clicker Question

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:

- Circuit-SAT
- Constraint satisfaction
- 3SAT
- Vertex Cover
- Set Cover
- Traveling Salesman
- 0/1 Knapsack
- Graph-Coloring
- Partitioning
- Numerical
- ... 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.