More NP-completeness

Recall how to prove a problem \( Z \) is NP-complete (after 1st proof):

1. Show \( Z \in \text{NP} \).
2. Show \( X \leq_p Z \) for some known NP-complete \( X \).

Road map for NP-completeness:

\[
\begin{align*}
\text{Circuit-SAT} & \leq_p \text{3-SAT} \quad \text{(pf. soon)} \\
\leq_p \text{3-SAT} & \leq_p \text{INDSET} \leq_p \text{VERTEX COVER} \\
\leq_p \text{INDSET} & \leq_p \text{DIRECTED HAM. CYCLE} \leq_p \text{HAM. CYCLE} \leq_p \text{TSP} \\
\leq_p \text{SUBSET SUM} & \leq_p \text{HAM. CYCLE} \leq_p \text{TSP} \\
\end{align*}
\]
Thm undirected Ham. cycle is NP-complete.

Pf. ① ∈ NP
② directed Ham. cycle ≤ₚ undirected Ham. cycle
   Given G = (V, E) input for directed ham. cycle,
   construct G', input for undirected " as follows

in G \[ \begin{array}{c}
\text{\begin{tikzpicture}
\node (v1) at (0,0) [circle, fill=black] {};
\node (v2) at (1,1) [circle, fill=black] {};
\node (v3) at (1,-1) [circle, fill=black] {};
\draw (v1) -- (v2);
\draw (v1) -- (v3);
\end{tikzpicture}}\end{array} \]

\[ \implies \begin{array}{c}
\text{\begin{tikzpicture}
\node (v1) at (0,0) [circle, fill=black] {};
\node (v2) at (1,1) [circle, fill=black] {};
\node (v3) at (1,-1) [circle, fill=black] {};
\draw (v1) -- (v2);
\draw (v1) -- (v3);
\end{tikzpicture}}\end{array} \]

in G'

Note: This is a many-one reduction (one-shot)

Claim 1 G has directed ham. cycle iff G' has ham. cycle

Claim 2 Poly. time.

Note: Here is an example to show why we need the extra vertex

G
\[ \begin{array}{c}
\text{\begin{tikzpicture}
\node (v1) at (0,0) [circle, fill=black] {};
\node (v2) at (1,1) [circle, fill=black] {};
\node (v3) at (1,-1) [circle, fill=black] {};
\draw (v1) -- (v2);
\draw (v1) -- (v3);
\end{tikzpicture}}\end{array} \]

no directed ham. cycle

G'
\[ \begin{array}{c}
\text{\begin{tikzpicture}
\node (v1) at (0,0) [circle, fill=black] {};
\node (v2) at (1,1) [circle, fill=black] {};
\node (v3) at (1,-1) [circle, fill=black] {};
\draw (v1) -- (v2);
\draw (v1) -- (v3);
\end{tikzpicture}}\end{array} \]

this has a ham. cycle

Note: This has a ham. cycle

no ham. cycle

Thm TSP (directed or undirected version) is NP-complete

Pf. Ex.
Thm Ham. path is NP-complete

Pf. Ex.
Subset Sum

Input: Numbers $w_1, \ldots, w_n$ and $W$

$Q$: Is there a subset $S \subseteq \{i \mid 1 \leq i \leq n\}$ such that $\sum_{i \in S} w_i = W$?

Summary:
- Dynamic programming alg., $O(n \cdot W)$ "pseudo-polynomial".
- Branch and bound alg., $O(2^n)$.

Thm: Subset Sum is NP-complete.

PF: $\exists \in \text{NP}$

1. $3$-SAT $\leq_p$ Subset Sum
   Assume we have a poly-time alg. for Subset Sum.
   Give a poly-time alg. for 3-SAT.

We've seen how to turn 3-SAT into a packing problem (ind. set) and into a sequencing problem (ham. cycle) and now we must turn it into a number problem.

Idea: specify the bits of the numbers

Given 3-SAT formula $F$ with clauses $C_1, \ldots, C_m$

Variables $x_1, \ldots, x_n$
Create a 0-1 matrix

\[
\begin{array}{cccc}
C_1 & C_2 & \cdots & C_m \\
\hline
x_1 & 1 & 0 & \quad \text{e.g.,} \\
-x_1 & 0 & 1 & C_1 = (x_1 \lor -x_2 \lor x_3) \\
x_2 & 0 & 0 & C_2 = (-x_1 \lor x_4 \lor x_5) \\
-x_2 & 1 & 0 & \text{general rule} \\
x_3 & 1 & 0 & M[i, j] = 1 \text{ if } C_j \text{ has } x_i \\
-x_3 & 0 & 0 & M[i, j] = 1 \text{ if } C_j \text{ has } -x_i \\
\vdots & \vdots & \vdots & \vdots \\
\end{array}
\]

we assume that no clause contains the same variable twice.

We will regard the rows as binary (or other base) numbers

\[
\begin{array}{cccc}
C_1 & C_2 & C_3 & \cdots \\
\end{array}
\]

Target sum \( \geq 1 \geq 1 \geq 1 \geq \cdots \)

issues:

1. we need some way to ensure we pick row \(x_i\) or row \(-x_i\) but not both
2. we need to handle target \( \geq 1 \)

What can the sum down a column be? \(1\) or \(2\) or \(3\)

Add slack of 1 or 2
Finally:

\[ W = \text{interpret bottom row in base 10} \]
\[ S = \text{one number for each row, interpreting the row in base 10} \]

so we have \(2n + 2m\) numbers of \(n + m\) digits.

Claim poly. time (and poly. size)
Claim $F$ is satisfiable iff $S$ has a subset with sum $W$

Pf. $\Rightarrow$ if $x_i$ is True, pick row $x_i$.
if $x_i$ is False, pick row $\neg x_i$.
Then column $c_j$ adds to 1, 2, or 3
Use slack rows $s_j^1, s_j^2$ to increase sum to 4
\[ 1 + s_j^1 + s_j^2 = 4 \]
\[ 2 + s_j^2 = 4 \]
\[ 3 + s_j^1 = 4 \]

This gives a set of rows (i.e., elements of $S$) with sum $W$.

$\Leftarrow$ Suppose $S$ has subset $S'$ with sum $W$
Note: any column sum is $\leq 6$ so no carries occur and column sums really must give target digit.
Because $x_i$ column sum is 1, we choose row $w$ $x_i$ or row $\neg x_i$ (not both) — set variable $x_i$ accordingly.
Because column $c_j$ sum is 4 and slacks sum to $\leq 3$, we must have chosen a literal to satisfy the clause $c_j$. 
The first NP-completeness proofs.

**Circuit Satisfiability**

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

i.e. \(x_1\) same as \(x_2\)

\[x_1 \equiv x_2\]

A circuit is a directed acyclic graph.

Sources (no edge entering) are labelled with variables or 0 or 1 — these are inputs.

Sink (no edge leaving) — there is one sink — output.

Internal nodes

A circuit computes an output when values are given to input variables.

e.g. above \(x_1 = 0\) \(x_2 = 1\) outputs 0

(compute values at internal nodes from sources to sink)
Circuit Satisfiability

Input: A circuit C.

Q: Is there an assignment of values to inputs s.t. output is 1? (Is C satisfiable?)

Thm Circuit SAT is NP-complete.

Pf. 1. Circuit SAT ∈ NP - easy
(2) (High-level idea) We must prove:

For every problem \( X \in \text{NP} \), \( X \leq_p \text{Circuit SAT} \)
i.e., for every problem \( X \in \text{NP} \) there is a poly-time alg. to transform any input \( I \) for \( X \) into a circuit \( C \) s.t. \( C \) is satisfiable iff \( I \) is a YES input for \( X \).
(Thus a poly-time alg. for Circuit SAT yields a poly-time alg. for \( X \)).

What can we use? Just that \( X \in \text{NP} \),
i.e., there is a poly-time verification alg. \( A \) for \( X \) that takes 2 inputs \( I, R \) and outputs YES/NO s.t.
\( I \) is a YES input for \( X \) iff
\( R \) s.t. \( \text{size}(R) \leq \text{poly. in size}(I) \)
\( \text{s.t. } A(I, R) \text{ outputs YES} \)

Idea: convert alg. \( A \) with known input \( I \) and unknown input \( R \) to a circuit \( C \) with input variables = bits of \( R \) s.t. \( C \) is satisfiable iff \( R \) s.t. \( A(I, R) \text{ outputs YES} \).

Program alg. \( A \). Compile, assemble ... at hardware level, this is implemented by \( \land, \lor, \neg \) gates. We get a circuit.

Inputs to circuit: bits of \( I \) (known)
bits of \( R \) (variables)
Internal nodes of circuit — memory locations after each time step of alg. A.

Because size(R) is poly. and A runs in poly. time, the circuit has poly. size.

Is there an alg. to convert A, I → c?
Yes, compiler, assembler etc. and poly. time.