More NP-completeness

Recall

How to prove a problem Z is NP-complete (after 1st proof)

① show $Z \in NP$

② show $X \leq_p Z$ for some known NP-complete X.

and recall our plan:

Circuit-SAT $\leq_p$ 3-SAT $\leq_p$ IND.SET $\leq_p$ VERTEX COVER $\leq_p$ SET COVER

3-SAT $\leq_p$ HAM.CYCLE $\leq_p$ TSP

3-SAT $\leq_p$ SUBSET SUM

pf. soon

today

X

Subset Sum

Input: Numbers $w_1 \cdots w_n$ and W input size $\sum \log w_i + \log W$

Q: Is there a subset $S \subseteq \{1 \cdots n\}$ s.t. $\sum_{i \in S} w_i = W$

Summary:

- dyn. prog. alg. $O(n \cdot W)$ "pseudo-poly."
- branch and bound alg. $O(2^n)$

Thm Subset Sum is NP-complete.

Pf ① $\in$ NP

② 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 $C_1 \cdots C_m$

variables $x_1 \cdots x_n$

Create a 0-1 matrix

	C_1	C_2	...	C_m
x_1	1	0		
$\neg x_1$	0	1		
x_2	0	0		
$\neg x_2$	1	0		
x_3	1	0		
$\neg x_3$	0	0		
⋮				

e.g.,
$C_1 = (x_1 \vee \neg x_2 \vee x_3)$
$C_2 = (\neg x_1 \vee x_4 \vee x_5)$

$M[x_i, C_j] = 1$ if x_i appears in C_j

$M[\neg x_i, C_j) = 1$ if $\neg x_i$ appears in C_j.

we assume that no clause contains the same variable twice.

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

	C_1	C_2	C_3	...
Target sum	≥ 1	≥ 1	≥ 1	...

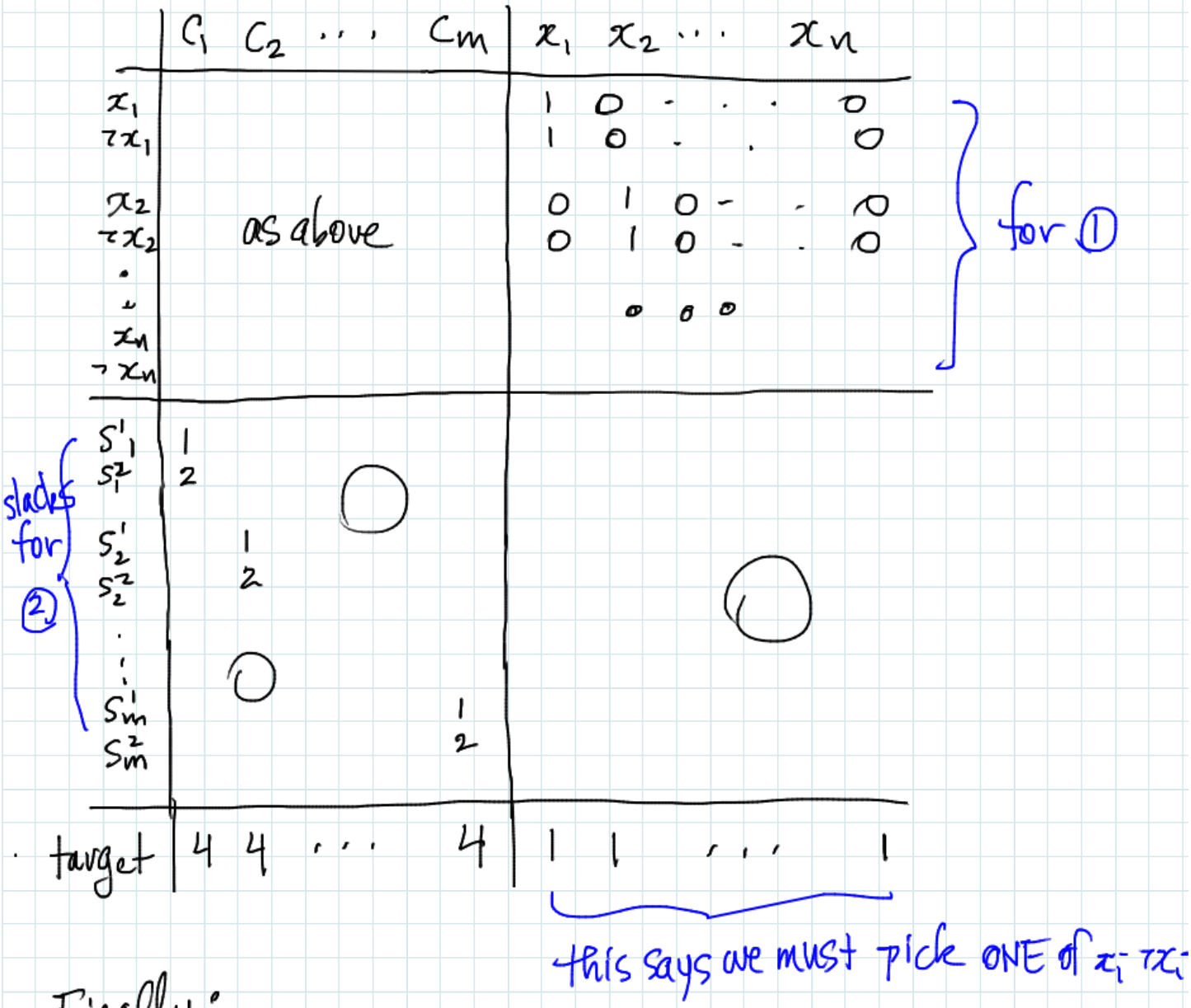
issues:

① we need some way to ensure we pick row x_i or row $\neg x_i$ but not both

② we need to handle target ≥ 1

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

Add <u>slack</u> of 1 or 2



Finally:

W = 'interpret bottom row in base 10

S = 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

$$1 + s_j^1 + s_j^2 = 4$$
$$2 + s_j^2 = 4$$
$$3 + s_j^1 = 4$$

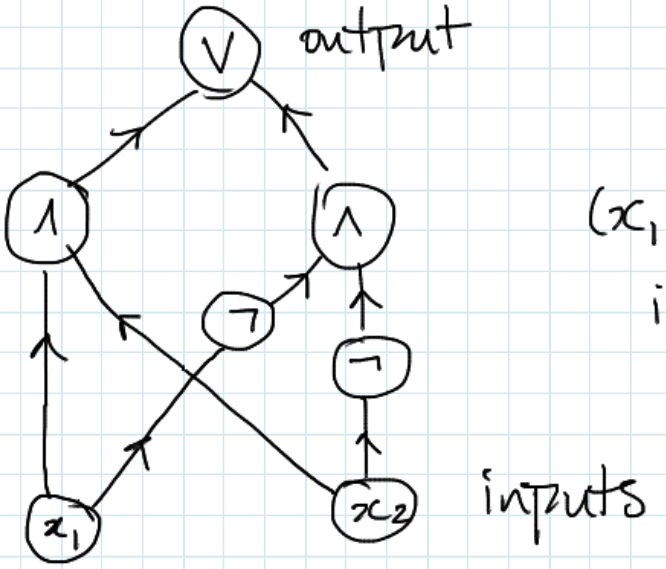
This gives a set of rows (i.e. elements of S) with sum W

$\Leftarrow$ Suppose S has subset S' with sum W
Note: any column sum is ≤ 6 so no carries occur and column sums really must give target digit
Because x_i column sum is 1, we choose row 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 ≤ 3, we must have chosen a literal to satisfy the clause c_j

Note: all our reductions for NP-completeness use the black-box subroutine only once and return its YES/NO answer.
You should always use this stronger reduction

OPEN: Are the 2 kinds of reduction equivalent inside NP?

The first NP-completeness proofs.
Circuit Satisfiability



$(x_1 \wedge x_2) \vee (\neg x_1 \wedge \neg x_2)$

i.e. x_1 same as x_2

$x_1 \equiv x_2$

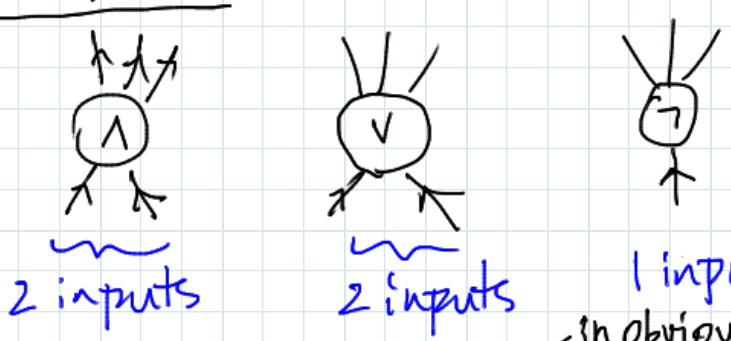
A <u>circuit</u> is a directed acyclic graph

<u>sources</u> (no edge entering) are labelled with variables or 0 or 1 — these are inputs

<u>sink</u> (no edge leaving)

there is one sink — output

<u>internal nodes</u>



A circuit <u>computes an output</u> (in obvious way) 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 $\in$ NP – easy

(2) (High-level idea) We must prove

for every problem $X \in NP$, $X \leq_p$ Circuit SAT

i.e. for every problem $X \in NP$ 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 $X \in NP$.

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

$\exists R$ size$(R) \leq$ poly. in size(I)

s.t. $A(I,R)$ 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 $\exists R$ s.t. $A(I,R)$ outputs YES.

program alg. A. Compile, assemble ...
at hardware level, this is implemented by $\wedge, \vee, \neg$ 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 to C?
Yes, compiler, assembler etc.
and poly. time.