Last time

$A \in NP \iff \exists$ TM V (verifier) with running time $t(n)$ that is polynomial in n s.t. $\forall x$

$x \in A \iff \exists y$ of length $t(|x|)$ s.t. V accepts $\langle x, y \rangle$

certificate

B is NP-hard iff $\forall A \in NP$, $A \leq_m^P B$

B is NP-complete iff (1) $B \in NP$
(2) B is NP-hard

Cook-Levin Theorem 3SAT is NP-complete

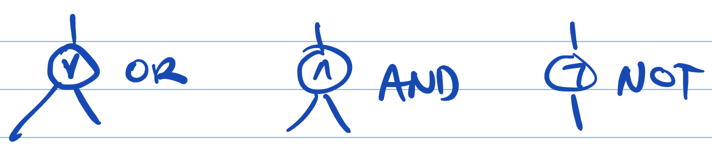
Proof: Already know $3SAT \in NP$

For NP-hardness we first show it for

CIRCUIT-SAT $= \{\langle C \rangle : C$ is a circuit and there is some input y s.t. $C(y) = 1\}$

Boolean circuit: inputs $x_1, \ldots, x_n$

gates: (bottom up)



Before we do this, we prove an interesting unrelated property of Boolean circuits.

Thm [Shannon] Almost all functions on n bits require circuit size $\Omega(2^n/n)$

Proof (Counting)

- How many functions on n bits are there?

  2^n bit vector inputs   2 choices for each

  total: 2^{2^n} functions

- How many circuits of size S are there on n inputs?

  S gates: • Each gate described by

  gate type   3 choices

  2 wires each input wire $S+n$ options

  other gates   input vars

  $\leq 3\cdot(S+n)^2$ options

  • Output gate S options

  Total: $S\cdot(3(S+n)^2)^S$

without loss of generality $S \geq n$

So $S^{O(S)}$ circuits of size $\leq S$.

More precisely $\leq S^{4S}$ circuits

(can actually get a better constant)

# of circuits of size $\leq 2^n/4n$ is

$$\leq \left(2^n/4n\right)^{4\cdot(2^n/4n)} = \left(\frac{2^n}{4n}\right)^{2^n/n} = \frac{2^{2^n}}{(4n)^{2^n/n}}$$

This $o(2^{2^n})$ so only a tiny fraction of functions. ∎

Thm CIRCUIT-SAT is NP complete

Proof (1) CIRCUIT-SAT $\in$ NP

Given $\langle C \rangle$:

certificate: String y for input assignment
length $\leq |\langle C \rangle|$ ✓

verify: Evaluate C on input y and accept iff value $= 1$
polytime ✓

(2) Show all $A \in NP$, $A \leq_m^P$ CIRCUIT-SAT

Let $A \in NP$

Goal: reduction f s.t.

$$A \qquad\qquad \text{CIRCUIT-SAT}$$

$$x \xmapsto{f} \langle C_{A,x} \rangle$$

↑ circuit depending on A and x

s.t. for all x: $x \in A \iff \exists y$ s.t. $C_{A,x}(y) = 1$

Since $A \in NP$
$\exists$ verifier V_A (1-tape TM)
s.t.
- V_A is polytime, say running time $T(n)$ that is $O(n^k)$

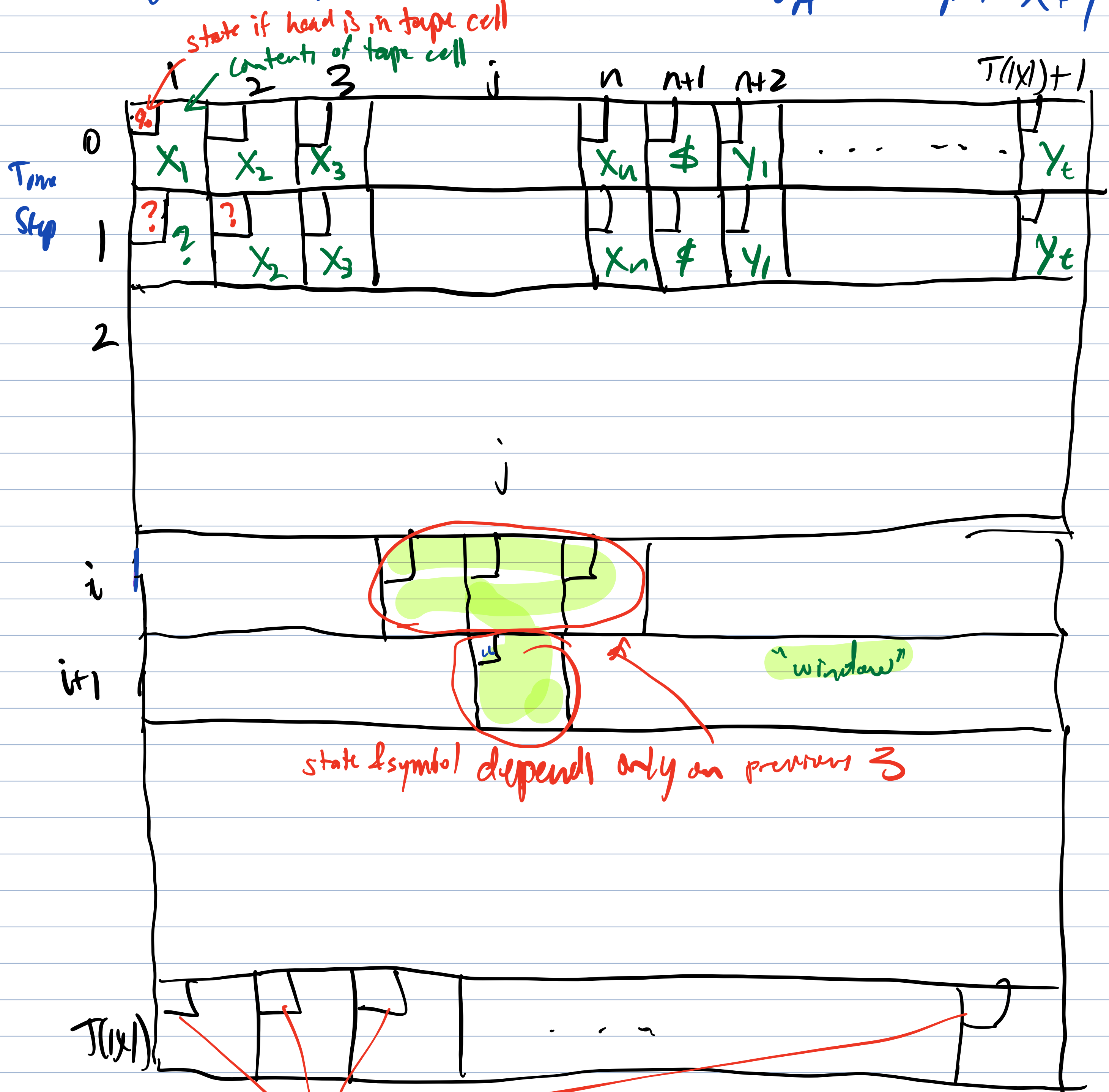
$\forall x \ \ x \in A \iff \exists y, |y|$ is $O(n^k)$ s.t. V_A accepts $\langle x,y \rangle$

Idea: create $C_{A,x}$ s.t. $C_{A,x}$ on input x should simulate V_A on input $\langle x,y \rangle$

w.l.o.g. $y \in \{0,1\}^*$ "bits"
and
$\langle x, y \rangle = x\$y$ $\quad \$ \notin \Gamma$

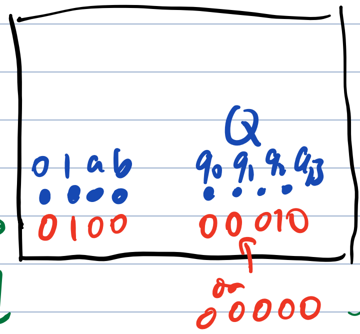
time bound for V_A
$|y| \le T(|x|) - |x|$ (any longer y wouldn't be looked at)

We now look at the "tableau" of V_A on input $x\$y$



accepts iff any of these is q_{accept}

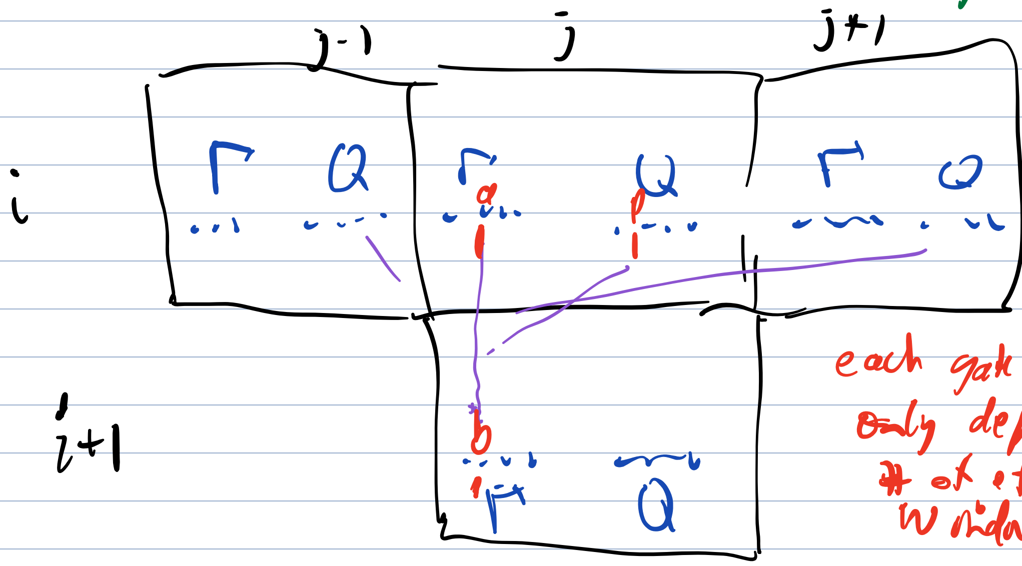
Representing each cell in a circuit:



one gate for each symbol in Γ
for each state in Q
"1-hot encoding"

exactly one gate evaluates to 1

at most one gate evaluates to 1.



Use δ function to build circuitry

each gate value only depends on a constant # of other values from the window

e.g. contents are b iff either

- Q gates are all 0's in cell above and cell above has b
- or Q gate for p in cell above has a 1, Γ gate for a in cell above has a 1 and $(p, a) = (q, b, R)$ or (q, b, L) for some $q \in Q$

(for some $p \in Q$, $a \in \Gamma$)

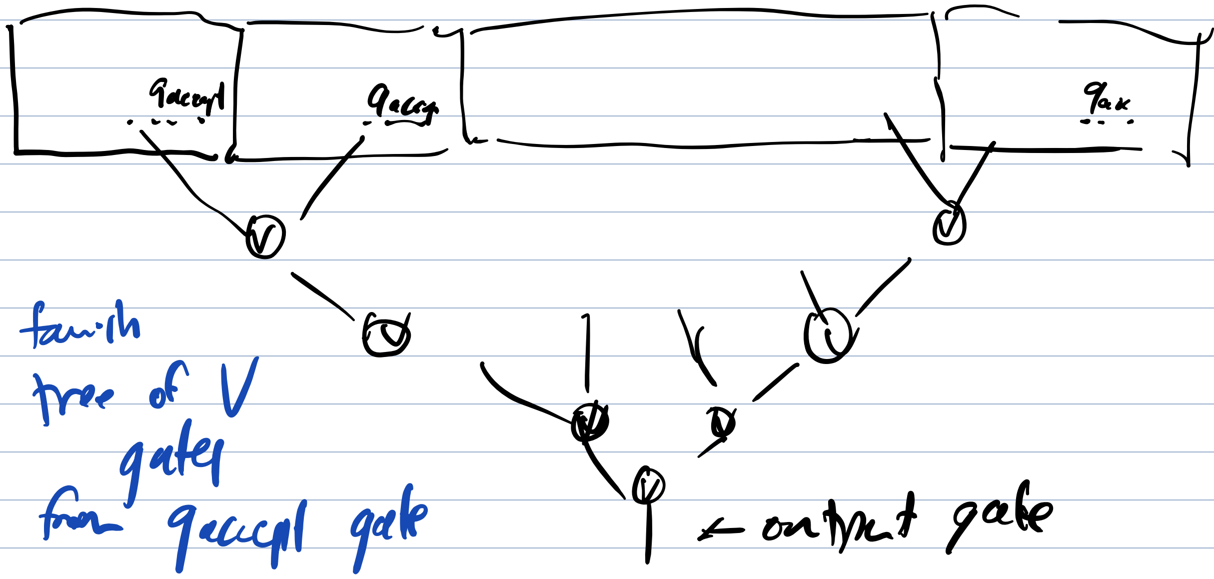
e.g. state is q iff for some $p \in Q$, $a \in \Gamma$, $b \in \Gamma$ either

- cell above and to left has gates for $p \in Q$ and $a \in \Gamma$ have value 1 and $\delta(p, a) = (q, b, R)$
- cell above and to right has gates for $p \in Q$ and $a \in \Gamma$ have value 1 and $\delta(p, a) = (q, b, L)$

The circuit $C_{A,x}$ has this same constant-sized piece repeated and linked in an entire grid of $(T/|x|+1) \times (T/|x|+1)$ cells
(slight change at left end)

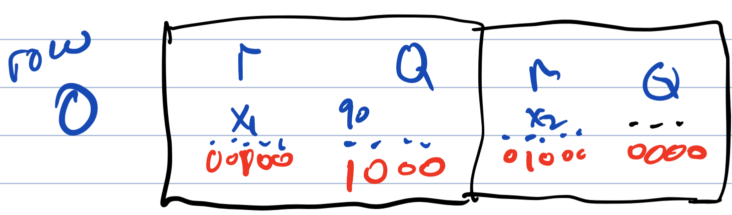
Output: We can assume wlog that q_{accept} values just get copied down to the bottom row (if they exist) as part of this circuit

Want output to be 1 iff V_A accepts $\langle x, y \rangle$
iff $\exists\, q_{accept}$ in final row

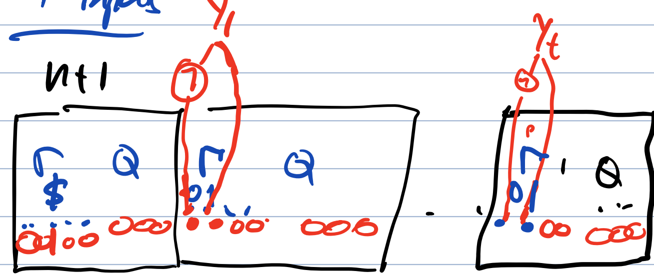


Big fan-in tree of $\vee$ gates from q_{accept} gate

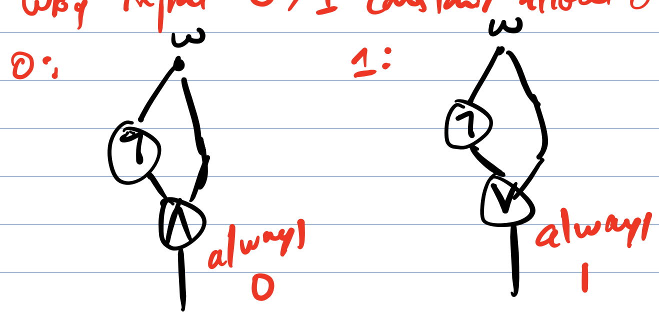
Inputs: $x\$y$:



circuit inputs



wlog input 0/1 constants allowed



Note: only symbols possible are 0 and 1

result: $0\,1\,\Gamma\,\dots$
$\neg y_i \; y_i \; 0\,0\,0$

Resulting circuit satisfies $C_{A,x}(y)=1$ iff V_A accepts x & y and is easy to compute:

Size for $|x|=n$ is $O(T^2(n))$ which is $O(n^{2k})$ polynomial.

$\therefore \forall A \in NP$, $A \leq_m^P$ CIRCUIT-SAT $\blacksquare$

We have now done the hard work. We first observe the following

Thm If $A \leq_m^P B$ and $B \leq_m^P C$ then $A \leq_m^P C$

Proof Let f be reduction showing $A \leq_m^P B$ time $O(n^k)$

" " g " reduction show $B \leq_m^P C$ time $O(n^\ell)$

$x \in A \Leftrightarrow f(x) \in B$ and $y \in B \Leftrightarrow g(y) \in C$

Let $h(x) = g(f(x))$. Then $x \in A \Leftrightarrow f(x) \in B$

Correctness $\Leftrightarrow g(f(x)) \in C$

$\Leftrightarrow h(x) \in C$

Running time: $O(|x|^k + |f(x)|^\ell)$ $\quad |f(x)|$ is $O(|x|^k)$

so $O(|x|^{k\ell})$ poly time $\blacksquare$

Thm C is NP-complete iff (1) $C \in NP$ (2) $B \leq_m^P C$ for some NP-complete B

Proof Given that $C \in NP$ only need to show C is NP-hard

By (2) $\forall A \in NP$, $A \leq_m^P B$ but then $A \leq_m^P B$ and $B \leq_m^P C \Rightarrow A \leq_m^P C$

so C is NP-hard too $\blacksquare$

We now prove

Thm 3SAT is NP-complete

Proof: 1. $3SAT \in NP$ ✓ prev. class

2. **Claim** $\text{CIRCUIT-SAT} \le_m^P 3SAT$

Want f: $\langle C \rangle \xmapsto{f} \langle \text{3CNF formula } \varphi \rangle$

s.t. C is SAT $\iff$ φ is SAT

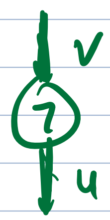
Now $C(y)=1 \iff \exists$ values for each gate of C consistent with input y such that output gate has value 1.

Design of φ: variables for y + variables for each gate of C

clauses
- represent constraints for gate values being correct
- say output value is 1

Note: gate values are carried on wires: we describe constraints for each gate type

NOT



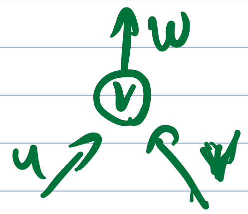
Want $\neg u \iff v$

i.e. $\neg u \rightarrow v$ i.e. $\neg\neg u \vee v$

$v \rightarrow \neg u$ etc.

Clauses
- $u \vee v$
- $\neg u \vee \neg v$

OR



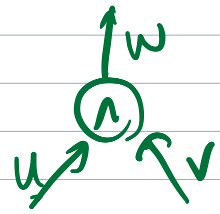
Want $w \Leftrightarrow (u \vee v)$

ie. $w \to (u \vee v)$

$(u \vee v) \to w$ ie. $u \to w$, $v \to w$

$$\boxed{\begin{array}{l} \text{Clauses: } \bullet\ \neg w \vee u \vee v \\ \bullet\ \neg u \vee w \\ \bullet\ \neg v \vee w \end{array}}$$

AND



Want $w \Leftrightarrow (u \wedge v)$

ie. $w \to u$, $w \to v$

$(u \wedge v) \to w$

$$\boxed{\begin{array}{l} \text{Clauses: } \bullet\ \neg w \vee u \\ \bullet\ \neg w \vee v \\ \bullet\ \neg u \vee \neg v \vee w \end{array}}$$

poly time (linear time) to compute.

Final formula has clauses like this for each gate plus clause of length 1 for output gate var.
Easy to compute. Clearly correct $\boxtimes$

Note: The formula above has ≤ 3 variables in each clause.

<u>Defn</u> EXACT-3SAT is like 3SAT but every clause has length $= 3$

<u>Thm EXACT 3SAT is NP-complete</u>

$3SAT \leq_m^p EXACT 3SAT$

Idea: for every clause of size 2

logically equivalent

$(a \vee b) \longmapsto (a \vee b \vee z)(a \vee b \vee \bar{z})$

for any variable z.

for clause of size 1:

logically equivalent

$a \mapsto (a \vee z_1 \vee z_2)(a \vee z_1 \vee \bar{z}_2)$
$(a \vee \bar{z}_1 \vee z_2)(a \vee \bar{z}_1 \vee \bar{z}_2)$
for any two vars z_1, z_2

$\blacksquare$

How to structure a proof that C is NP-complete

(1) <u>$C \in NP$</u>: For input x

(a) Give the form of <u>certificate</u> for x and argue <u>poly length</u>

(b) Give algorithm to <u>verify</u> certificate and argue <u>poly time</u>

and then

(2) <u>C is NP-hard</u>: Choose known NP-complete problem A and write

<u>Claim</u> $A \leq^P_m C$

want f s.t. $x \mapsto f(x)$
$x \in A \iff f(x) \in C$

(a) Define function f

(b) Argue f computable in polytime

(c) Correctness:

Argue $x \in A \iff f(x) \in C$.

Usually best argument is of following form since both $A, C \in NP$ and have polytime verifiers

(i) $x \in A \Rightarrow f(x) \in C$:

Since $x \in A$ there is a certificate y for $x \in A$

input $x \longmapsto f(x)$

certificate $y \qquad y'$

Show how to use y to build certificate y' for $f(x) \in C$

(ii) $f(x) \in C \Rightarrow x \in A$

do the reverse. Given certificate

note only need y' for special form $f(x)$ not general inputs to C → (y'' for $f(x) \in C$ show how to get

get certificate y''' for $x \in A$

∎

Correctness example:

CIRCUIT-SAT $\qquad$ 3SAT

$\langle C \rangle \longmapsto \langle \varphi \rangle$

certificates ⇒ y assignment s.t. $C(y)=1$ $\longmapsto y' = (y, z)$ $\quad$ z = gate values on input y

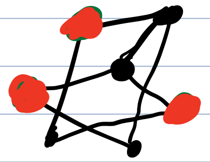
⇐ just take input values in y'' ⟵ Given y'' making φ true

Easy reduction: $3SAT \leq_m^P CNFSAT$: f = identity (if input not of right form map to garbage)

INDEPENDENT-SET $= \{\langle G, k\rangle :$ G is a graph with an independent set of size $\geq k\}$

where

Def^n For a graph $G=(V,E)$, $I \subseteq V$ is *independent* iff no pair of vertices in I joined by an edge



Thm INDEPENDENT-SET is NP-complete

Proof 1) $\in$ NP: **Certificate** for G: set of vertices U forming an independent set of size k (length $\leq$ length of encoding of the graph) so poly size

Verify Check that
- no edges between elt^s of I
- $|I| \geq k$.

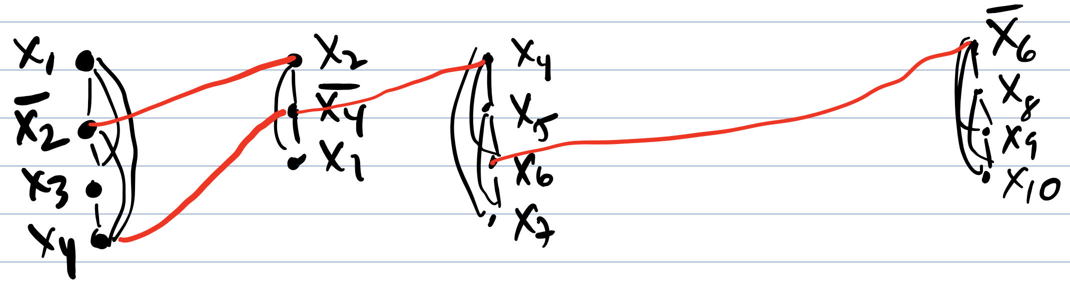
both easily polytime

(2) **NP-hard**: **Claim** CNFSAT $\leq_m^P$ INDEP-SET
3SAT

Let φ be a CNF formula with clauses $C_1, \ldots, C_m$ on n vars.

(a) Define G with one vertex per literal occurrence in φ, organized in columns for each clause eg $C_1 = (x_1 \vee \bar{x_2} \vee x_3 \vee x_4)$. $C_2 =$)

G: C_1 C_2 C_3 C_4 ... C_m



Put an edge between every pair of vertices in same column

$\Rightarrow$ independent set has size $\leq m$

Put an edge between every pair of nodes labelled by contradictory literals $x_i, \bar{x_i}$

G has both kinds of edges

map: $\langle \varphi \rangle \xmapsto{f} \langle G, m \rangle$

(b) f is clearly polytime G has # of vertices $\leq$ size of $\langle \varphi \rangle$ and # of edges at most that squared.

easy to compute

(c) Correctness:

(i) Suppose φ is satisfiable with assignment y making it 1

y must make every clause $C_1 .. C_m$ true i.e. make at least one literal in each clause true

For set I choose one of these true literals per clause/column (doesn't matter which)

That won't contain any black edge because I has at most one literal per column

It won't contain any red edge since y can't make both $x_i, \bar{x}_i$ true.

$\therefore$ I is independent & size m as required.

(ii) Suppose G has an independent set I' of size m

I' must have one node per column

For truth ass y' set all the literals labelling these nodes to true.

(This will be consistent because I' can't contain nodes labelled both $x_i, \bar{x}_i$ because of red edges)

This might leave some vars unassigned so far by y'; assign these remaining vars arbitrarily in y'

Clearly this y' will satisfy φ by construction ∎