Lecture 21

Hw 5 is out!

- Do the $\in$ NP & (3) : they are "easy"
- NP-hard is more challenging

A verifier V is a deterministic algorithm

$L = \{ w \mid V \text{ accepts } \langle w, c \rangle \text{ for some string } c \}$

A verifier is polytime if it runs in polynomial time in $|w|$ => not $|w| + |c|$

c is a "certificate"

$NP = \mathcal{L}_1 = \{L \mid L$ is decided by a polytime NTM$\}$

$NP = \mathcal{L}_2 = \{L \mid L$ has a polytime verifier$\}$

Thm 7.20

Last time we started to show the equivalence between both defs

Ie: polytime NTM $\Leftrightarrow$ polytime Verifier

↓ nondeterministic

↓ deterministic

but extra string

c

We showed $\mathcal{L}_1 \subseteq \mathcal{L}_2$

Still need to show $\mathcal{L}_2 \subseteq \mathcal{L}_1$

Let $L \in \mathcal{L}_2$ w.t.s $L \in \mathcal{L}_1$

Let V be a verifier for L running in time n^k.

Nondeterministic TM for L

On input x

poly-time
- Nondeterministically pick a string c of length n^k
- Run $V(x, c)$ | polytime
- If V accepts: accept else

reject

If $w \in L$, $\exists c$ such $V(x, c) =$ accept

when we guess this c the NTM will accept

If $w \notin L$, $\forall c : V(x, c) =$ reject

So the NTM never accepts

$SAT = \{ \langle \phi \rangle \mid \phi$ is a satisfiable CNF

formula }

Input is a formula over variables $x_1, \dots x_n$

A clause is an OR of litterals

↳ L_i $\overline{x_i}$, x_i

↳ $C = (x_1 \vee \overline{x_2} \vee x_7)$

A CNF is of the form

$C_1 \wedge C_2 \wedge \dots C_m$

Ex: $\phi = (x_1 \vee \overline{x_2} \vee x_3) \wedge (x_3 \vee x_4)$

$\wedge (\overline{x_2} \vee \overline{x_1}) \wedge (\overline{x_2} \vee x_3)$

set $x_1 = 1$, $x_2 = 0$, $x_3 = 1$, $x_4 = 1$

Def: ϕ is satisfiable if $\exists$ a 0/1 assignment to the variables ($\alpha \in \{0,1\}^n$) such that $\phi(\alpha) = 1$

↳ For every clause C_i of ϕ is true under α

↳ For each clause at least 1 literal is set to 1 under α.

If α set x_i to 0 then $x_i = 0$

$\overline{x_i} = 1$

A special case: 3-SAT = $\{\langle \phi \rangle \mid \phi$ is a satisfiable CNF formula but every clause has at most 3 litterals$\}$

$(x_1 \vee x_2 \vee \bar{x}_3)$ ✓

$(x_1 \vee x_2)$ ✓

$(x_1 \vee x_2 \vee x_3 \vee x_4)$ ✗

To encode a CNF formula with n variables & m clauses => write all m clauses as an $2n$ bit string

x_1	$\bar{x}_1$	x_2	$\bar{x}_2$	.....	x_n	$\bar{x}_n$
1	0	1	0	----	1	0

=> $x_1 \vee x_2 \vee x_n$

if that literal is in the clause

Can encode using $O(n \cdot m)$ bits

SAT & 3SAT are in NP!

If n variables : 2^n assignments possible

2^n isn't poly $(n \cdot m)$

Verifier for 3SAT:

On input $\langle \phi, c \rangle$:

$O(n)$ (- check that $c \in \{0,1\}^n$) an assignment to the variables

m times (- check that for each clause C_i:

$poly(n)$ | c satisfies C_i) set some litteral in C_i to true

$O(1)$ (- If all clauses are satisfied: accept.

- Else reject.

Running time: $O(n) + O(m \cdot poly(n))$

$= poly(n \cdot m)$

$= poly(|\phi|)$

IF $\phi \in$ 3-SAT, $\exists$ some c that makes $V(\phi, c)$ accept.

$c \leftarrow$ the assignment that satisfies ϕ

IF $\phi \notin$ 3-SAT: $\forall c$ $V(\phi, c)$ rejects

↳ Then for all c, c must falsify some clause of ϕ. So V always reject.

$P = NP$?

NP completeness: Levin (Toni's PhD advisor)

& Cook showed some problems in NP are as hard as all NP-problems (NP-hard)

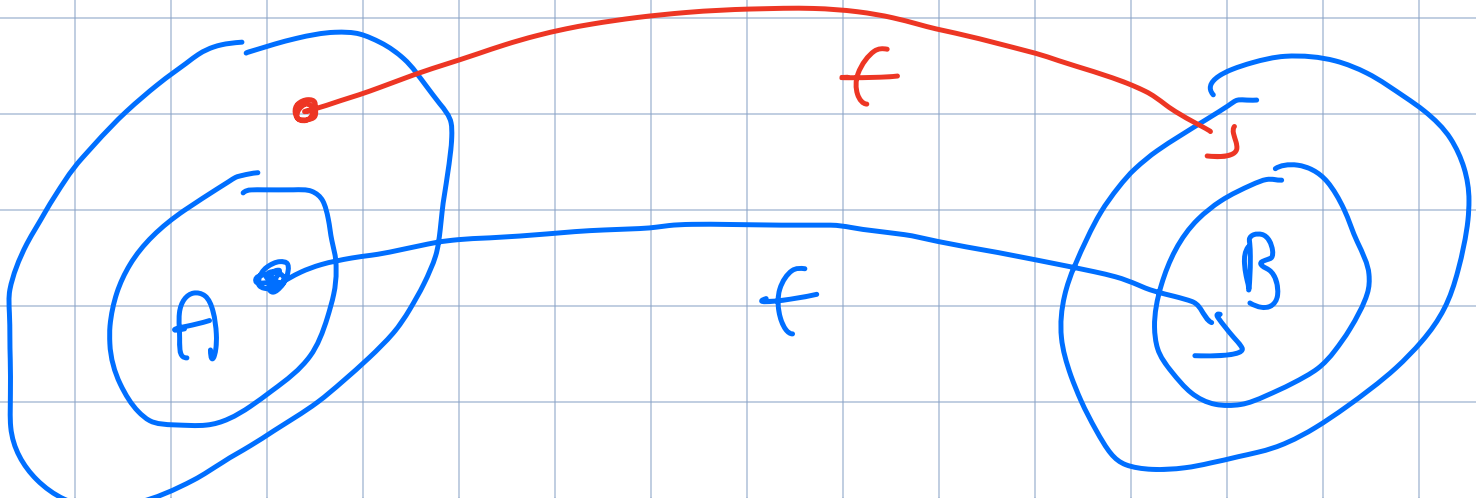
Ex: If we could solve 3-SAT in polytime we could solve all problems in NP in poly time

I.e: If 3-SAT $\in$ P $\Rightarrow$ P = NP

We say 3-SAT is NP-hard

Definition: Language A is polytime (mapping) reducible to a language B ($A \leq_p B$) if $\exists$ a polynomial-time computable function $f: \Sigma^* \to \Sigma^*$ such that

$$w \in A \iff f(w) \in B$$



Idea: If B can be solved in polytime. To solve A in polytime

- On input x
- Compute $f(x)$ | polytime
- check if $f(x) \in B$ | polytime
- If $f(x) \in B \Rightarrow w \in A$

  $f(x) \notin B \Rightarrow w \notin A$

Definition:

- A language B is NP-hard if

$$\forall A \in NP \; : \; A \leq_p B$$

↳ idea: If $B \in P \Rightarrow P = NP$

- B is NP-complete if

B is NP-hard AND

B is in NP

Thm: [Cook, Levin] 3-SAT is NP hard

L_3 since 3-SAT $\in$ NP

3 SAT is NP complete

Now to show L is NP hard.

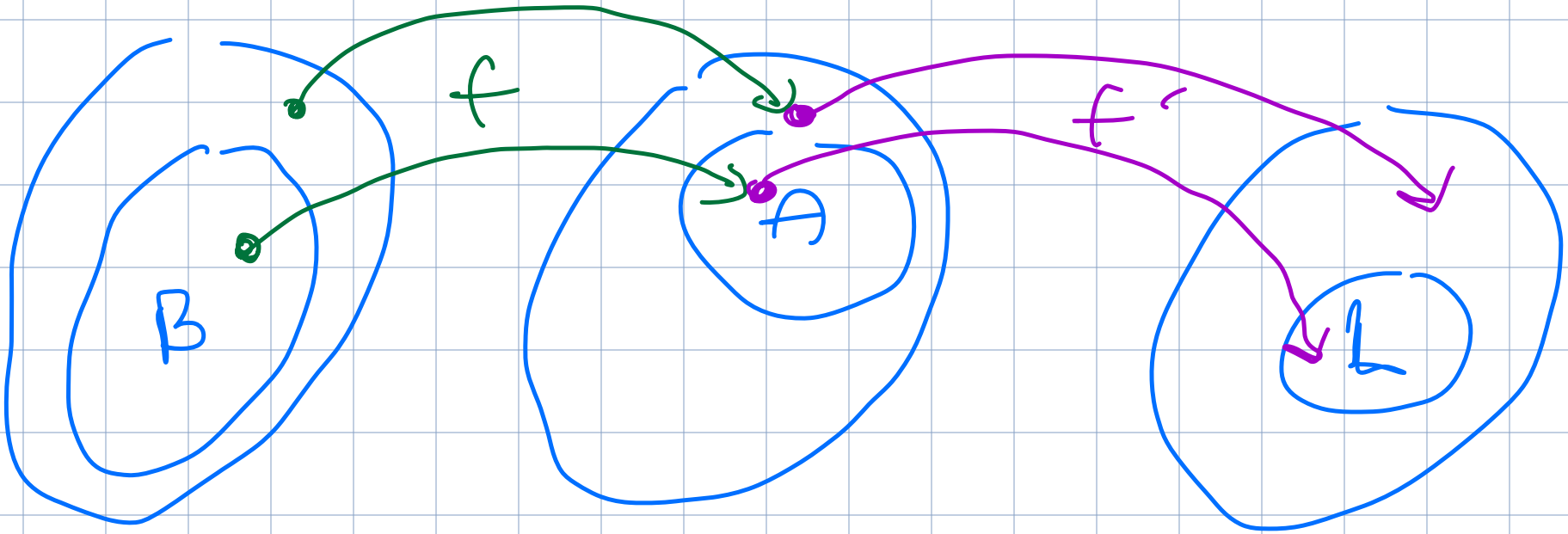
Show $A \leq_p L$ where A is NP hard

By def: $\forall B \in NP$, $B \leq_p A$

But $A \leq_p L$

$\Rightarrow B \leq_p L$ $\forall$ languages B

$\in NP$



Thm: Clique is NP-complete

Clique = $\{ \langle G, k \rangle \mid G$ has a k-clique$\}$

(1) CLIQUE $\in NP$ ✓

(2) Clique is NP-hard

↳ Clique is NP complete

We will show $3SAT \leq_p Clique$

Given $\langle \phi \rangle$ need to output $\langle G, k \rangle$ in polynomial time such that

$$\langle \phi \rangle \in 3SAT \iff \langle G, k \rangle \in Clique$$

On input $\langle \phi \rangle$ we will output

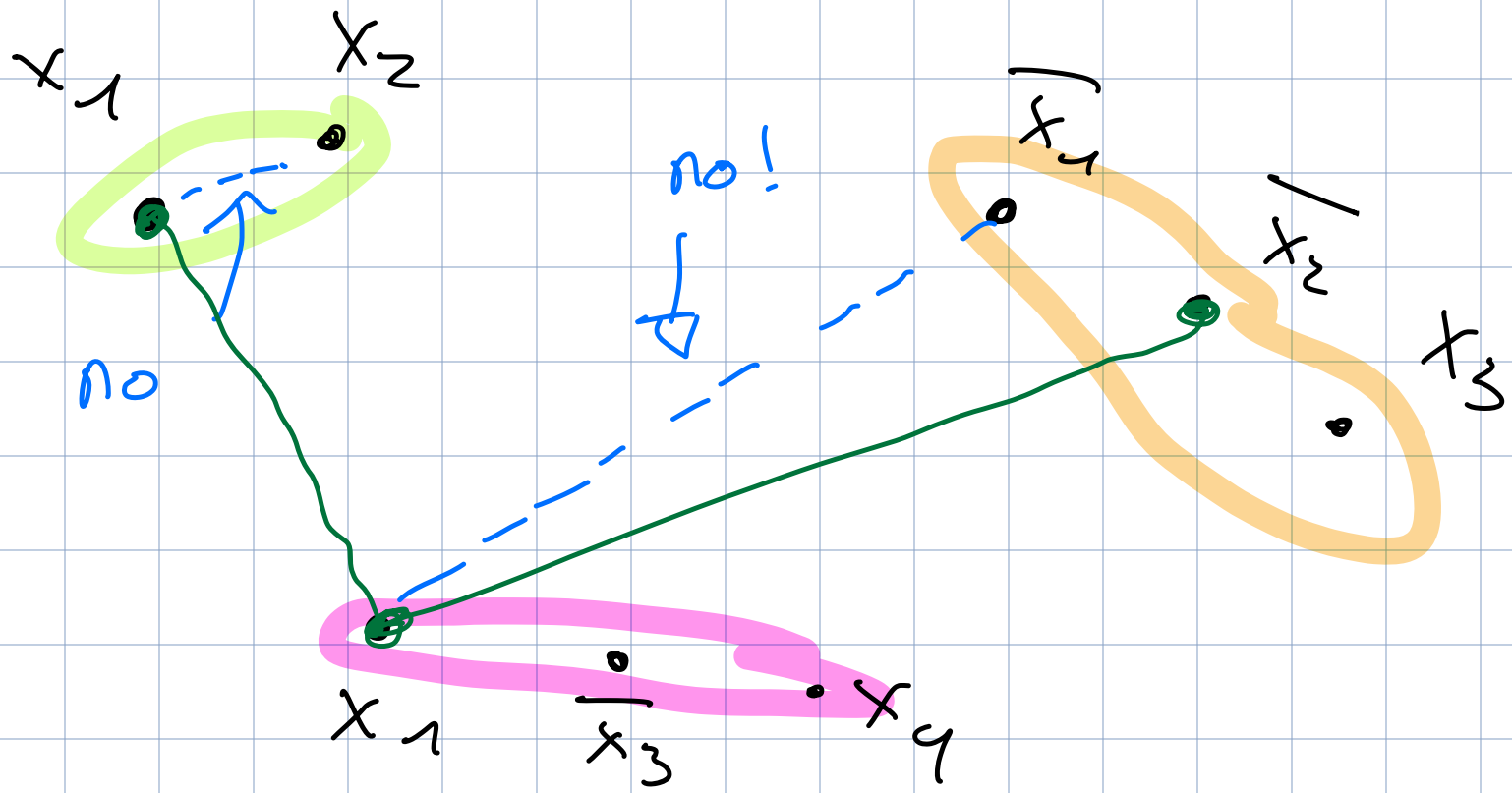
$\langle G_\phi, m \rangle$

G_ϕ: a graph depending on ϕ

m: # of clauses in ϕ

What is G_ϕ?

$$\phi = (x_1 \vee x_2) \wedge (\bar{x}_1 \vee \bar{x}_2 \vee x_3) \wedge (x_1 \vee \bar{x}_3 \vee x_4)$$



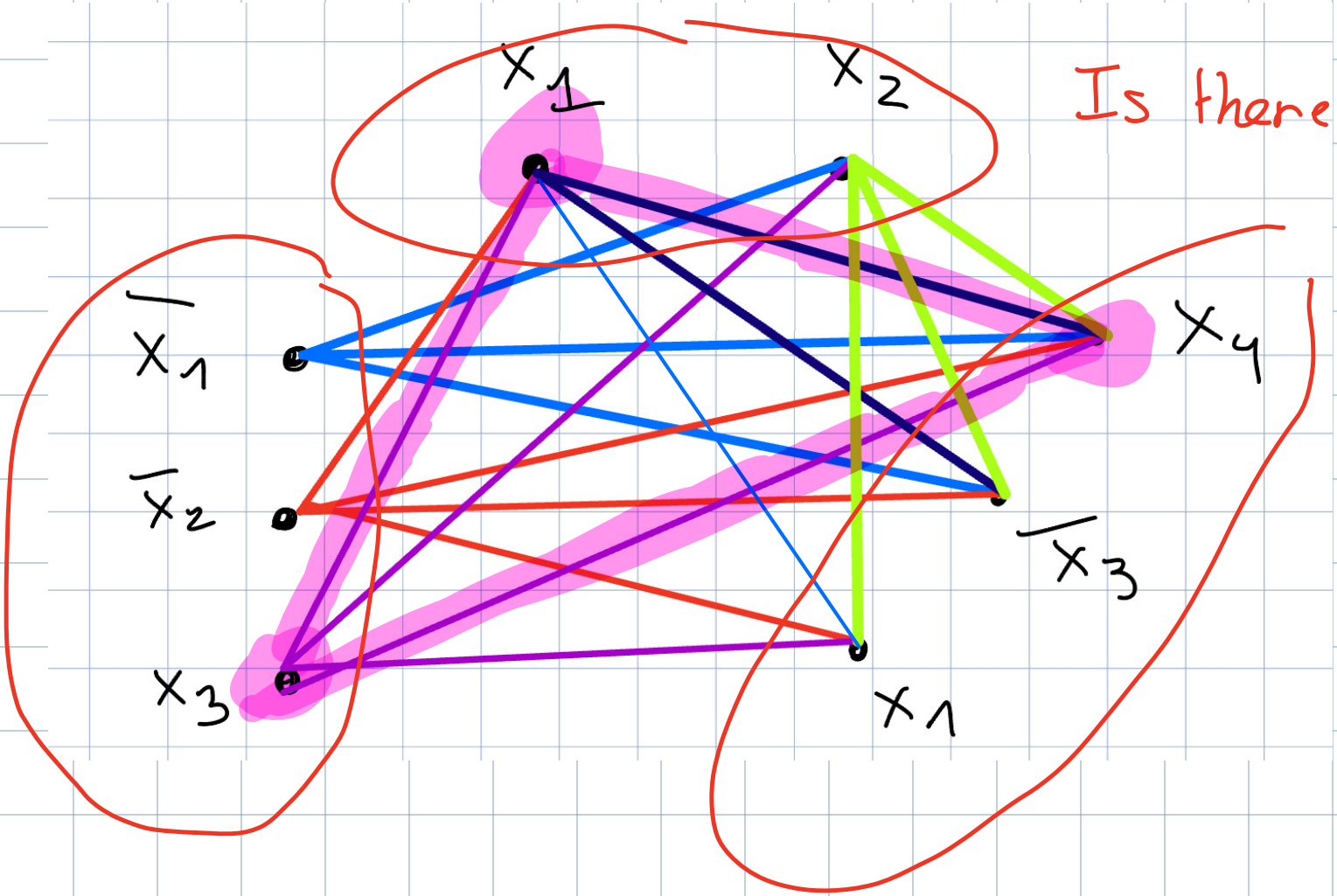
add edges : add an edge between vertices

(u, v) will be an edge iff:

u & v are in different clusters

v & u are not opposite litterals

ie $v = x_i$, $u = \overline{x_i}$



Is there a clique of size m ?

$$\phi = (x_1 \vee x_2) \wedge (\overline{x_1} \vee \overline{x_2} \vee x_3) \wedge (x_1 \vee \overline{x_3} \vee x_4)$$

$x_1 = 1$ $x_2 = 0$ $x_3 = 1$ $x_4 = 1$

Formal proof:

On input ϕ we compute $f(\langle\phi\rangle)$ as follows:

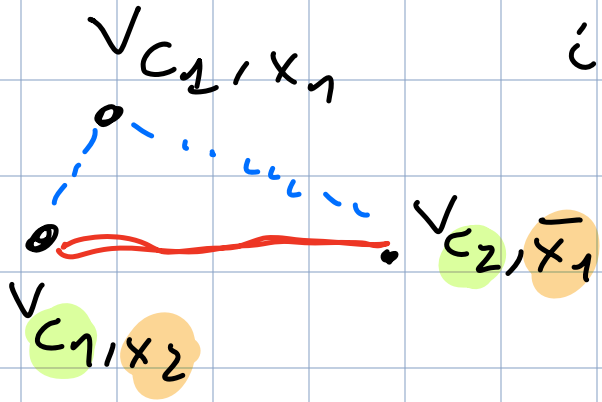
Build the graph G_ϕ as follows:

(1) For each clause $C = \ell_1 \vee \ell_2 \vee \ell_3$ of ϕ =s create vertices

$$v_{C,\ell_1}, v_{C,\ell_2}, v_{C,\ell_3}$$

(2) Add edges:

Add an edge between $v_{C,\ell}$ & $v_{C',\ell'}$



if $c \neq c'$ and ℓ, ℓ' are not opposite litterals

Output $\langle G_\phi, m \rangle$

1) Running time to compute f

(1) takes poly(n) time per clause

So poly(n, m) time in total

(2) To add edges $\Rightarrow$ at most $\leq n \cdot m$ vertices so poly$(n \cdot m)$

$\leq 3 \cdot m$ time to build all the edges

So computing takes poly time.

Need to show ϕ is satisfiable iff G_ϕ has a clique of size m.

1) ϕ is satisfiable $\hookrightarrow$ $\langle G_\phi, m \rangle \in$ Clique

Let a be a satisfying assignment for

ϕ.

For each clause C, pick a vertex $v_{C,\ell}$ where ℓ is satisfied by α.

Then this set of m vertices forms a clique of size m in G_ϕ.

→ all vertices come from a different clause, they can't have opposite littorals

→ all edges are present between

these vertices

2) If G_ϕ has a clique of size m then ϕ is satisfiable.

Let S be a clique of size m in G_ϕ.

↳ For each clause C exacly one

$$v_{C,\ell} \in S$$

Never picked 2 vertices with opposite litterals.

So let α be the following assignement:

- For each variable x_i:
  - Set $x_i = 1$ if we have some vertex with $\ell = x_i$
  - Set $x_i = 0$ if we have some vertex with $\ell = \overline{x}_i$
  - Else set $x_i = 1$ (default)

Punch line: α satisfies ϕ:

For each clause C, it sets ℓ

to true where l is the sectional

of $V_{c,l}$ in S