18.404/6.840 Lecture 15

Last time:

- NTIME$\big(t(n)\big)$, NP
- P vs NP problem
- Dynamic Programming, $A_{\text{CFG}} \in$ P
- Polynomial-time reducibility

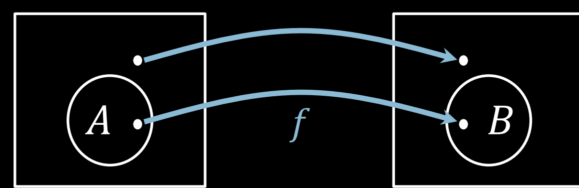
Today: (Sipser §7.5)

- NP-completeness

Quick Review

Defn: A is polynomial time reducible to B $(A \leq_{\mathrm{P}} B)$ if $A \leq_{\mathrm{m}} B$ by a reduction function that is computable in polynomial time.

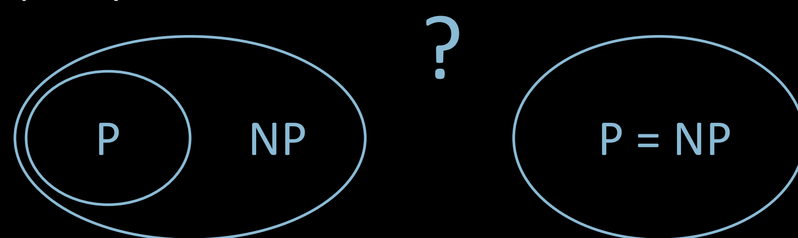
Theorem: If $A \leq_{\mathrm{P}} B$ and $B \in$ P then $A \in$ P.



f is computable in polynomial time

NP = All languages where can verify membership quickly
P = All languages where can test membership quickly

P versus NP question: Does P = NP?



$SAT = \{\langle\phi\rangle \mid \phi$ is a satisfiable Boolean formula$\}$

Cook-Levin Theorem: $SAT \in$ P $\rightarrow$ P = NP
Proof plan: Show that every $A \in$ NP is polynomial time reducible to SAT.

$\leq_\mathrm{P}$ Example: $3SAT$ and $CLIQUE$

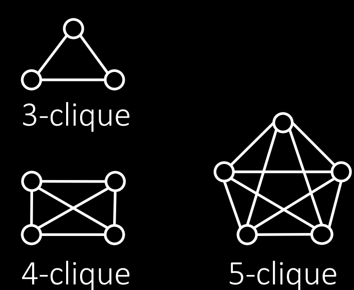
Defn: A Boolean formula ϕ is in Conjunctive Normal Form (CNF) if it has the form $\phi = \underbrace{(x \vee \overline{y} \vee z)}_{\text{clause}} \wedge \underbrace{(\overline{x} \vee \overline{s} \vee z \vee u)}_{\text{clause}} \wedge \cdots \wedge (\overline{z} \vee \overline{u})$

literals

Literal: a variable or a negated variable
Clause: an OR ($\vee$) of literals.
CNF: an AND ($\wedge$) of clauses.
3CNF: a CNF with exactly 3 literals in each clause.
$3SAT = \{\langle\phi\rangle \mid \phi \text{ is a satisfiable 3CNF formula}\}$

Defn: A k-clique in a graph is a subset of k nodes all directly connected by edges.
$CLIQUE = \{\langle G, k\rangle \mid \text{graph } G \text{ contains a } k\text{-clique}\}$

Will show: $3SAT \leq_\mathrm{P} CLIQUE$



$3SAT \leq_P CLIQUE$

Theorem: $3SAT \leq_P CLIQUE$

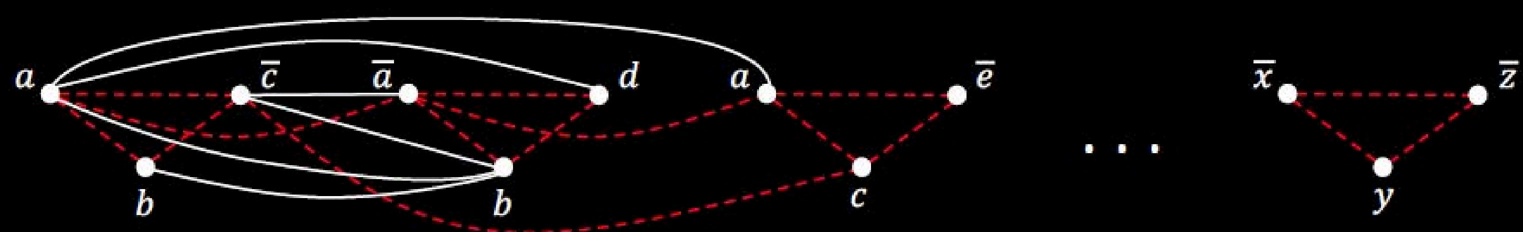
Proof: Give polynomial-time reduction f that maps ϕ to G, k where ϕ is satisfiable iff G has a k-clique.

A satisfying assignment to a CNF formula has ≥1 true literal in each clause.

$$\phi \ = \ (a \vee b \vee \overline{c}) \wedge (\overline{a} \vee b \vee d) \wedge (a \vee c \vee \overline{e}) \wedge \ \cdots \ \wedge (\overline{x} \vee y \vee \overline{z})$$

$\big\downarrow f$

$G =$



$k =$ # clauses

Forbidden edges:

1) within a clause
2) inconsistent labels (a and $\overline{a}$)

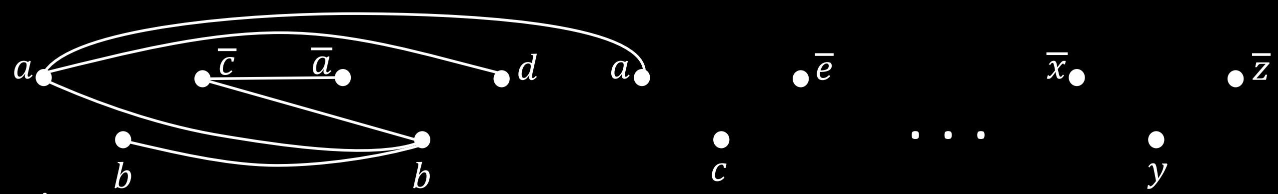
G has <u>all</u> non-forbidden edges

$3SAT \leq_{\mathrm{P}} CLIQUE$ conclusion

$\phi = (a \vee b \vee \overline{c}) \wedge (\overline{a} \vee b \vee d) \wedge (a \vee c \vee \overline{e}) \wedge \cdots \wedge (\overline{x} \vee y \vee \overline{z})$

$\downarrow f$

$\boxed{\begin{matrix} G \\ k \end{matrix}}$ $\begin{matrix} = \\ = \text{\# clauses} \end{matrix}$



Claim: ϕ is satisfiable iff G has a k-clique

(→) Take any satisfying assignment to ϕ. Pick 1 true literal in each clause.
The corresponding nodes in G are a k-clique because they don't have forbidden edges.

(←) Take any k-clique in G. It must have 1 node in each clause.
Set each corresponding literal TRUE. That gives a satisfying assignment to ϕ.

The reduction f is computable in polynomial time.

Corollary: $CLIQUE \in \mathrm{P} \rightarrow 3SAT \in \mathrm{P}$

Check-in 15.1

Does this proof require 3 literals per clause?

(a) Yes, to prove the claim.

(b) Yes, to show it is in poly time.

(c) No, it works for any size clauses.

Check-in 15.1

NP-completeness

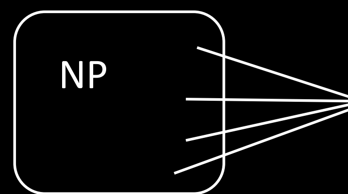
Defn: B is <u>NP-complete</u> if

1) $B \in \mathrm{NP}$
2) For all $A \in \mathrm{NP}$, $A \leq_{\mathrm{P}} B$

If B is NP-complete and $B \in \mathrm{P}$ then $\mathrm{P} = \mathrm{NP}$.

Cook-Levin Theorem: SAT is NP-complete

Proof: Next lecture; assume true



next lecture $\leq_{\mathrm{P}} SAT$

today $\leq_{\mathrm{P}} HAMPATH$

To show some language C is NP-complete, show $3SAT \leq_P C$.

or some other previously shown NP-complete language

Check-in 15.2

What language that we've previously seen is most analogous to SAT?

(a) A_{TM}

(b) E_{TM}

(c) $\{0^k 1^k \mid k \geq 0\}$

Check-in 15.2

$HAMPATH$ is NP-complete

Theorem: $HAMPATH$ is NP-complete

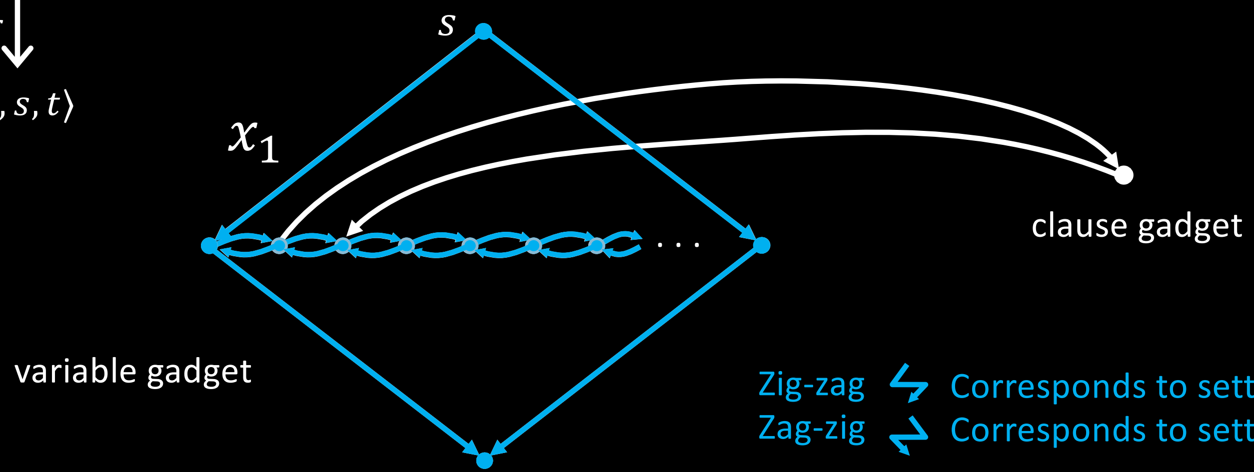
Proof: Show $3SAT \leq_P HAMPATH$ (assumes $3SAT$ is NP-complete)

Idea: "Simulate" variables and clauses with "gadgets"

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

$f \downarrow$

$\langle G, s, t \rangle$



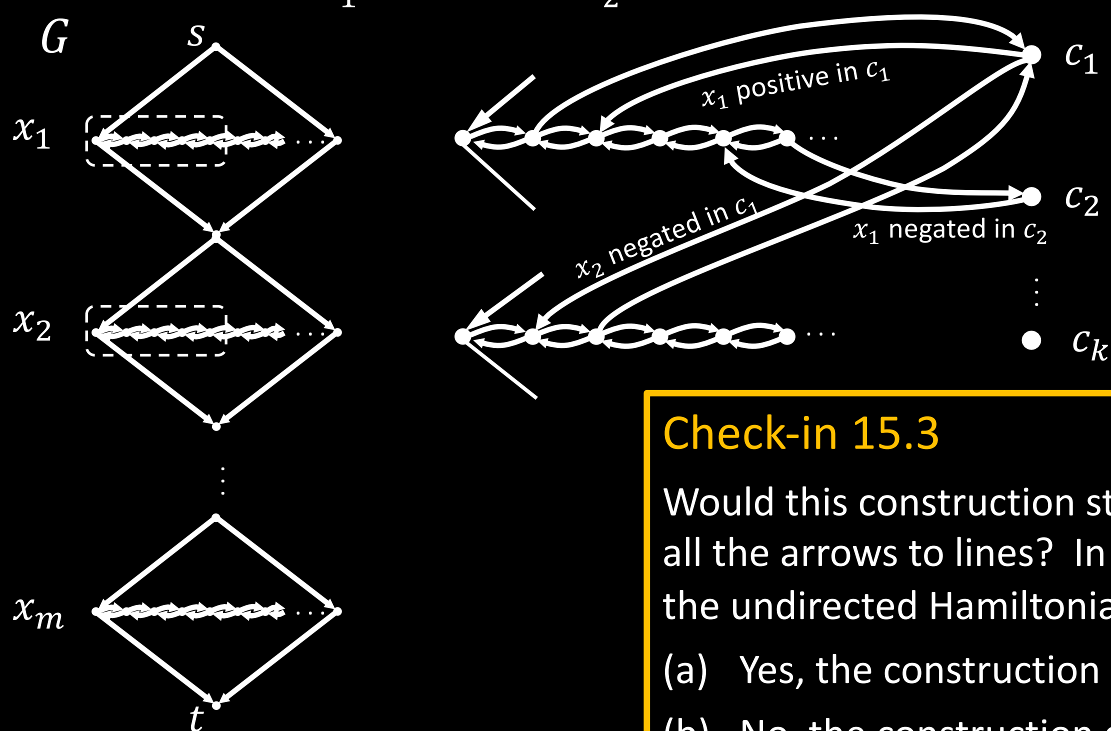
Zig-zag Corresponds to setting x_1 TRUE

Zag-zig Corresponds to setting x_1 FALSE

Construction of G

$$\phi = \underbrace{(x_1 \vee \overline{x}_2 \vee x_3)}_{c_1} \wedge \underbrace{(\overline{x}_1 \vee x_2 \vee x_4)}_{c_2} \wedge \cdots \wedge \underbrace{(\qquad x_m \;)}_{c_k}$$

m variables
k clauses



The reduction f is computable in polynomial time.

Check-in 15.3

Would this construction still work if we made G undirected by changing all the arrows to lines? In other words, would this construction show that the undirected Hamiltonian path problem is NP-complete?

(a) Yes, the construction would still work.

(b) No, the construction depends on G being directed.

Check-in 15.3

Quick review of today

1. NP-completeness
2. SAT and $3SAT$
3. $3SAT \leq_P HAMPATH$
4. $3SAT \leq_P CLIQUE$
5. Strategy for proving NP-completeness: Reduce from $3SAT$ by constructing gadgets that simulate variables and clauses.

18.404J / 18.4041J / 6.840J Theory of Computation
Fall 2020