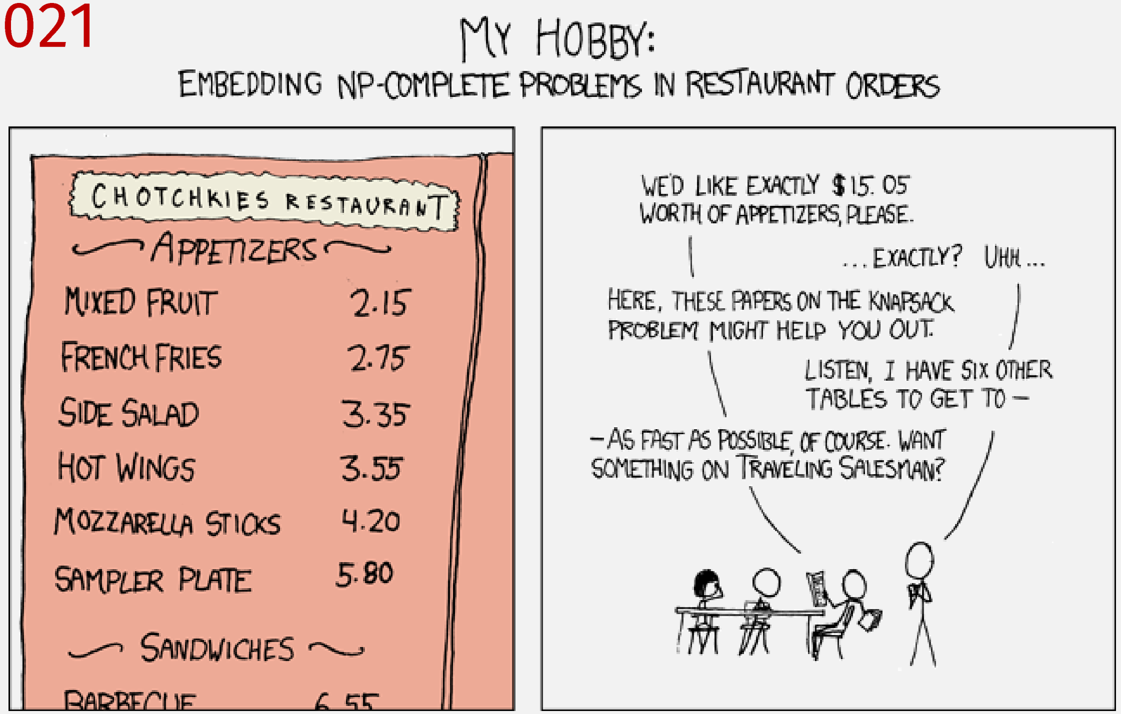
More NP-Complete Problems

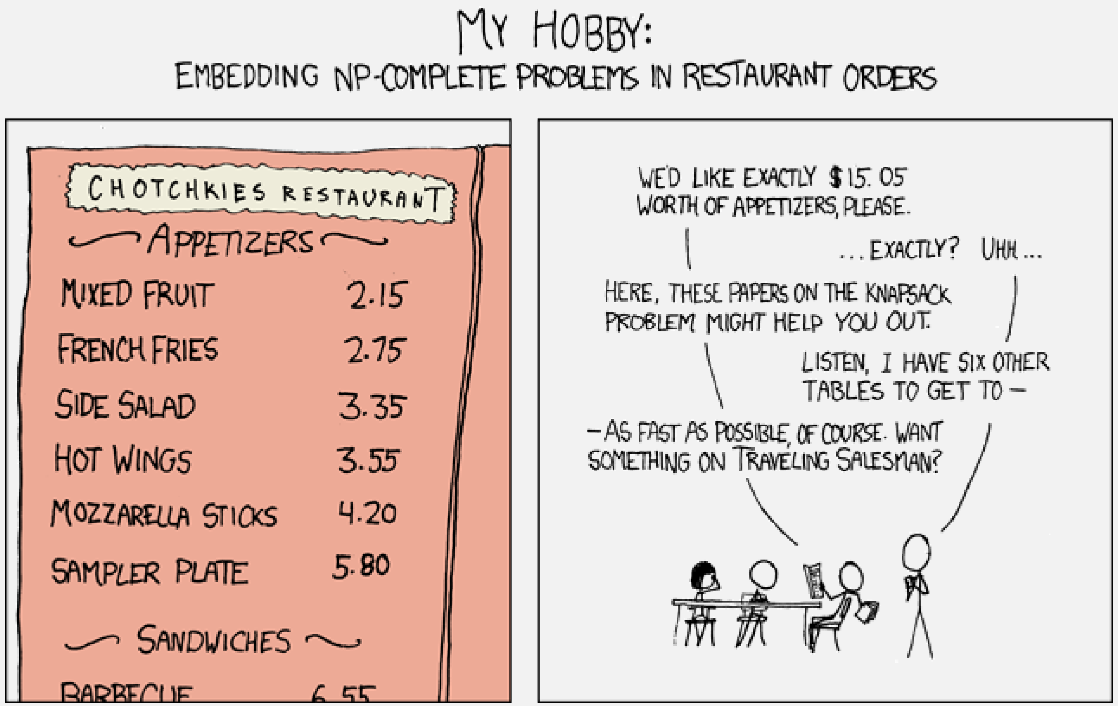
Wed, May 5 2021



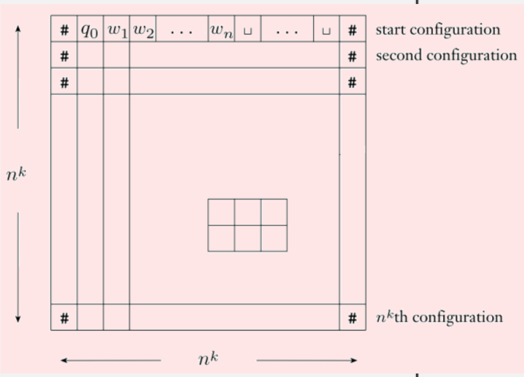
Announcements

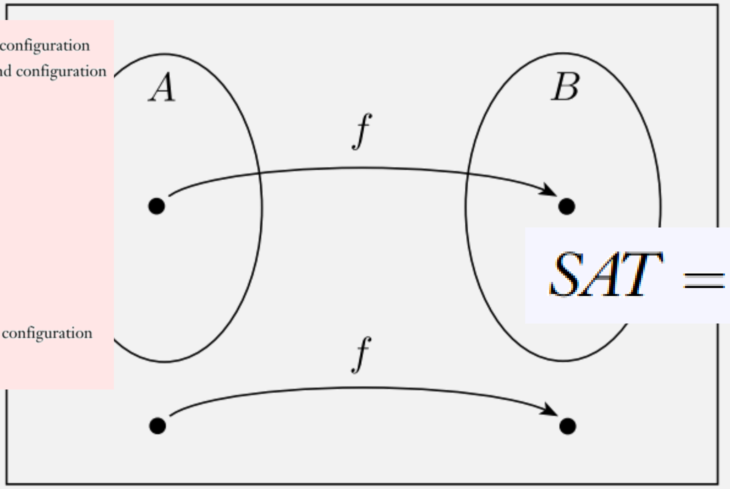
- HW11 due 11:59pm EST tonight

- HW12 out
  - Due Wed 5/12 11:59pm EST
  - Last hw



Last Time: SAT is NP-complete





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

$$\phi_{\text{cell}} \wedge \phi_{\text{start}} \wedge \phi_{\text{move}} \wedge \phi_{\text{accept}}$$

Now it will be much easier to prove that other languages are NP-complete

THEOREM 7.36

known / unknown

Key Thm: If B is NP-complete and $B \leq_P C$ for C in NP, then C is NP-complete.

To use this theorem, C must be in **NP**

Proof:

- Need to show: C is **NP**-complete, i.e., (Def 7.34):
  - it's in **NP** (given), and
  - every lang A in **NP** reduces to C in poly time (must show)
- For every language A in **NP**, reduce $A \rightarrow C$ by:
  - First reduce $A \rightarrow B$ in poly time
    - Can do this because B is **NP**-Complete
  - Then reduce $B \rightarrow C$ in poly time
    - This is given
- Total run time: Poly time + poly time = poly time

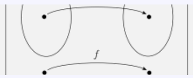
THEOREM 7.36 known unknown

Using: If B is NP-complete and $B \leq_P C$ for C in NP, then C is NP-complete.

To use this theorem, must know *C* is in **NP**

Example: Prove *3SAT* is **NP**-Complete using thm 7.36 …

- … by constructing poly time reduction from:
    - $SAT = \{\langle\phi\rangle \mid \phi \text{ is a satisfiable Boolean formula}\}$ (known to be **NP**-Complete)

      to 
    - $3SAT = \{\langle\phi\rangle \mid \phi \text{ is a satisfiable 3cnf-formula}\}$ (known to be in **NP**)

THEOREM 7.36

Using: If B is NP-complete and $B \leq_P C$ for C in NP, then C is NP-complete.

Example: Prove *3SAT* is **NP**-Complete using thm 7.36 ...

- ... by constructing poly time reduction from:
  - $SAT = \{\langle\phi\rangle |\ \phi$ is a satisfiable Boolean formula$\}$ (known to be **NP**-Complete)

    to
  - $3SAT = \{\langle\phi\rangle |\ \phi$ is a satisfiable 3cnf-formula$\}$ (known to be in **NP**)
- Reduction: Given an arbitrary *SAT* formula:
  1. Convert to conjunctive normal form (CNF), ie an AND of OR clauses
     - Use DeMorgan's Law to push negations onto literals O(**n**)

       $\neg(P \vee Q) \iff (\neg P) \wedge (\neg Q) \qquad \neg(P \wedge Q) \iff (\neg P) \vee (\neg Q)$
     - Distribute ORs to get ANDs outside of parens O(**n**)

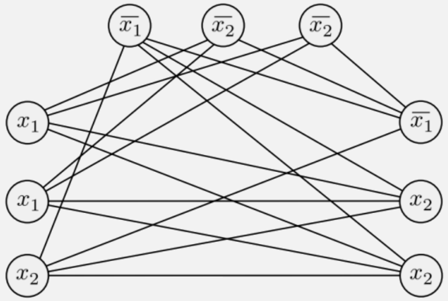
       $(P \vee (Q \wedge R)) \Leftrightarrow ((P \vee Q) \wedge (P \vee R))$
  2. Then split clauses to 3cnf by adding new variables O(**n**)

     $(a_1 \vee a_2 \vee a_3 \vee a_4) \qquad (a_1 \vee a_2 \vee z) \wedge (\overline{z} \vee a_3 \vee a_4)$

Remaining Step (on your own): show iff relation holds

NP-Complete problems, so far

- $SAT = \{\langle\phi\rangle \mid \phi \text{ is a satisfiable Boolean formula}\}$ (Cook-Levin Theorem)
- $3SAT = \{\langle\phi\rangle \mid \phi \text{ is a satisfiable 3cnf-formula}\}$ (just now: reduced *SAT* to *3SAT*)
- $CLIQUE = \{\langle G, k\rangle \mid G \text{ is an undirected graph with a } k\text{-clique}\}$
  - *CLIQUE* is in NP (Thm 7.24)
  - *3SAT* is polynomial time reducible to *CLIQUE*. (Thm 7.32)

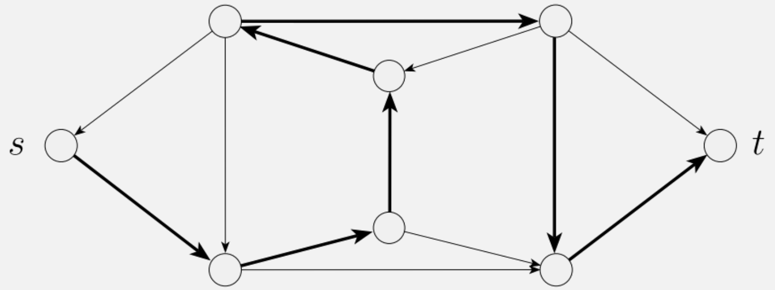


THEOREM 7.36

If B is NP-complete and $B \leq_P C$ for C in NP, then C is NP-complete.

Other **NP** (not shown complete yet) Problems, so far

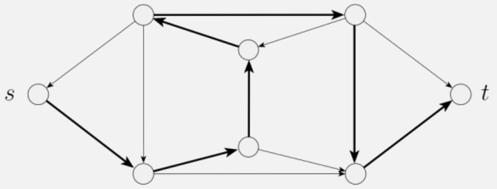
- $HAMPATH = \{\langle G, s, t\rangle \mid G \text{ is a directed graph with a Hamiltonian path from } s \text{ to } t\}$
  - A Hamiltonian path goes through every node in the graph



All **NP**-Complete!
(will prove it today)

- $SUBSET\text{-}SUM = \{\langle S, t\rangle \mid S = \{x_1, \ldots, x_k\}, \text{ and for some } \{y_1, \ldots, y_l\} \subseteq \{x_1, \ldots, x_k\}, \text{ we have } \Sigma y_i = t\}$
  - Some subset of a set of numbers sums to some total
  - e.g., $\langle \{4, 11, 16, 21, 27\}, 25\rangle \in SUBSET\text{-}SUM$

Theorem: *HAMPATH* is NP-complete



$HAMPATH = \{\langle G, s, t\rangle |\ G$ is a directed graph with a Hamiltonian path from s to $t\}$

THEOREM 7.36

If B is NP-complete and $B \leq_P C$ for C in NP, then C is NP-complete.

Strategy: Use

Proof Parts (5):

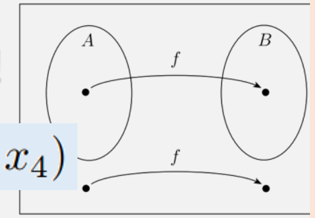
☑ 1. Show *HAMPATH* is in **NP** (done in prev class)

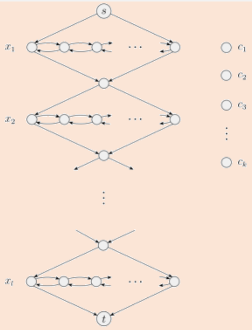
☑ 2. Choose **NP**-complete problem to reduce from: *3SAT*

3. Create the computable function:

Coming up next!

$(x_1 \vee \overline{x_2} \vee \overline{x_3}) \wedge (x_3 \vee \overline{x_5} \vee x_6) \wedge (x_3 \vee \overline{x_6} \vee x_4)$





DEFINITION 7.29

Language A is ***polynomial time mapping reducible***,[1] or simply ***polynomial time reducible***, to language B, written $A \leq_P B$, if a polynomial time computable function $f: \Sigma^* \longrightarrow \Sigma^*$ exists, where for every w,

$$w \in A \Longleftrightarrow f(w) \in B.$$

4. Show it runs in poly time

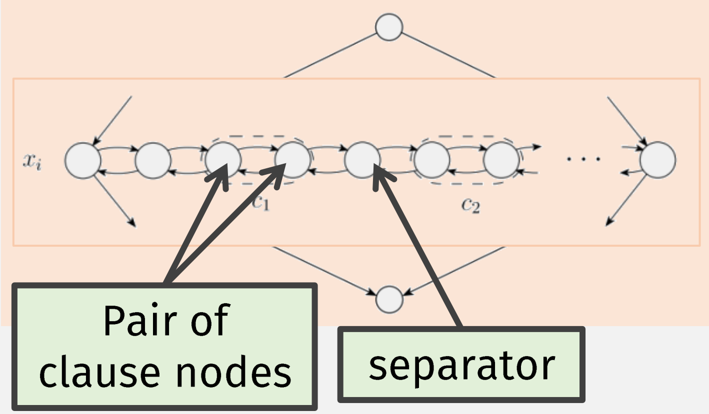
5. Show Def 7.29 "iff" requirement:
   - Satisfiable 3cnf formula ⇔ graph with Hamiltonian path

Computable Fn: Formula (blue) → Graph (orange)

Example input: $\phi = (a_1 \vee b_1 \vee c_1) \wedge (a_2 \vee b_2 \vee c_2) \wedge \cdots \wedge (a_k \vee b_k \vee c_k)$

k = # clauses

- Clause → (extra) single nodes, Total = k
- Variable → diamond-shaped graph "gadget"
  - Clause → 2 "connector" nodes + separator
  - Total = $3k+1$ "connector" nodes per "gadget"

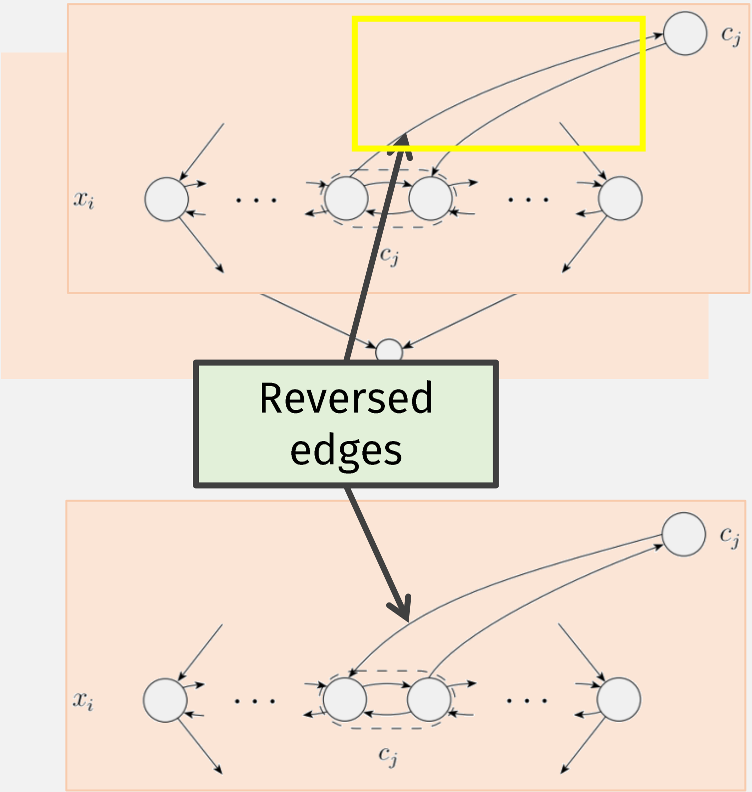


Computable Fn: Formula (blue) → Graph (orange)

Example input: $\phi = (a_1 \vee b_1 \vee c_1) \wedge (a_2 \vee b_2 \vee c_2) \wedge \cdots \wedge (a_k \vee b_k \vee c_k)$

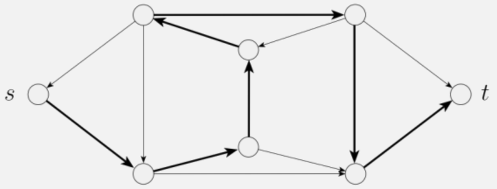
k = # clauses

- Clause → (extra) single nodes, Total = k
- Variable → diamond-shaped graph "gadget"
  - Clause → 2 "connector" nodes + separator
  - Total = $3k$+1 "connector" nodes per "gadget"



- Lit x_i in clause c_j → c_j node edges in gadget x_i
- Lit $\overline{x_i}$ in clause c_j → c_j edges in gadget x_i (rev)

Theorem: *HAMPATH* is NP-complete



HAMPATH $= \{\langle G, s, t\rangle \mid G$ is a directed graph with a Hamiltonian path from s to $t\}$

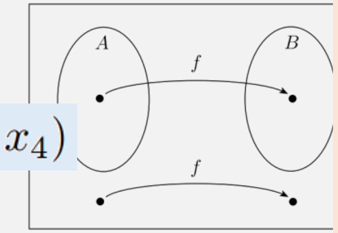
Proof Parts (5):

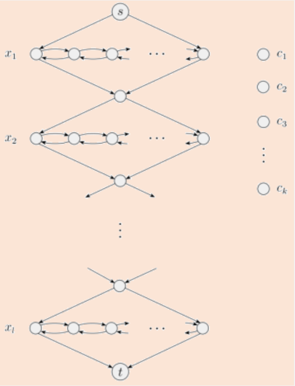
☑ 1. Show *HAMPATH* is in **NP** (done in prev class)

☑ 2. Choose **NP**-complete problem to reduce from: *3SAT*

☑ 3. Create the computable function:

$(x_1 \vee \overline{x_2} \vee \overline{x_3}) \wedge (x_3 \vee \overline{x_5} \vee x_6) \wedge (x_3 \vee \overline{x_6} \vee x_4)$





4. Show it runs in poly time
5. Show Def 7.29 iff requirement:
   - Satisfiable 3cnf formula ⇔ graph with Hamiltonian path

Polynomial Time?

TOTAL: O($\boldsymbol{k}^2$)

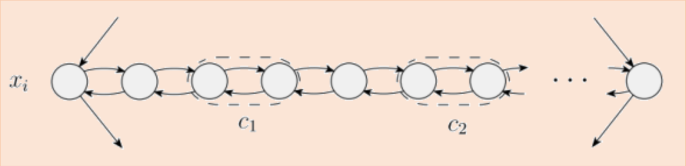
Example input: $\phi = (a_1 \vee b_1 \vee c_1) \wedge (a_2 \vee b_2 \vee c_2) \wedge \cdots \wedge (a_k \vee b_k \vee c_k)$

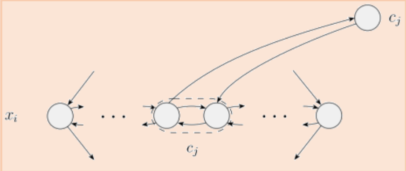
k = # clauses = at most $\mathbf{3}\boldsymbol{k}$ variables

- Clause → (extra) single nodes 

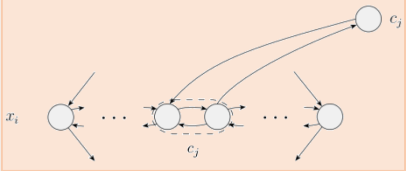
O($\boldsymbol{k}$)

- Variable → diamond-shaped graph "gadget" O($\boldsymbol{k}^2$)
  - Clause → 2 "connector" nodes + separator
  - Total = $3k$+1 "connector" nodes per "gadget"



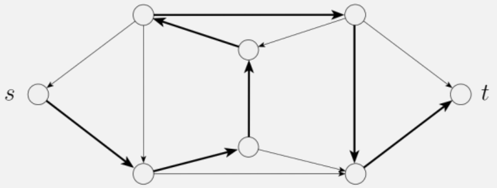
- Lit x_i in clause c_j → c_j node edges in gadget x_i 

O($\boldsymbol{k}$)

- Lit $\overline{x_i}$ in clause c_j → c_j edges in gadget x_i (rev) 

O($\boldsymbol{k}$)

Theorem: *HAMPATH* is NP-complete



$$HAMPATH = \{\langle G, s, t\rangle \mid G \text{ is a directed graph with a Hamiltonian path from } s \text{ to } t\}$$

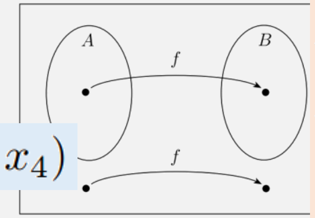
Proof Parts (5):

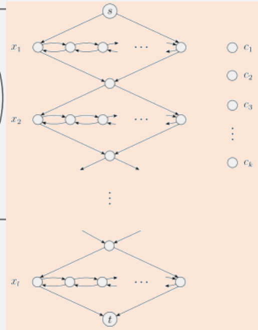
☑ 1. Show *HAMPATH* is in **NP** (done in prev class)

☑ 2. Choose **NP**-complete problem to reduce from: *3SAT*

☑ 3. Create the computable function:

$$(x_1 \vee \overline{x_2} \vee \overline{x_3}) \wedge (x_3 \vee \overline{x_5} \vee x_6) \wedge (x_3 \vee \overline{x_6} \vee x_4)$$





DEFINITION 7.29

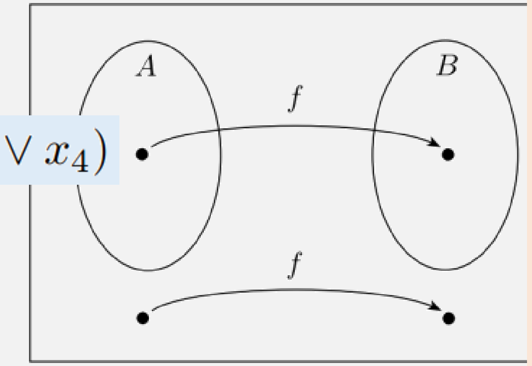
Language A is ***polynomial time mapping reducible***,[1] or simply ***polynomial time reducible***, to language B, written $A \leq_P B$, if a polynomial time computable function $f\colon \Sigma^* \longrightarrow \Sigma^*$ exists, where for every w,

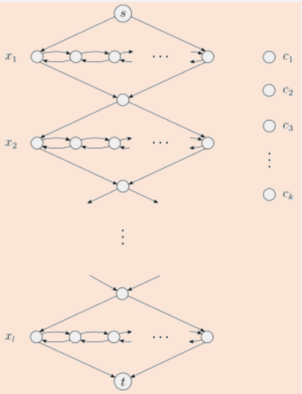
$$w \in A \iff f(w) \in B.$$

☑ 4. Show it runs in poly time

5. Show Def 7.29 iff requirement:
   - Satisfiable 3cnf formula ⇔ graph with Hamiltonian path

$(x_1 \vee \overline{x_2} \vee \overline{x_3}) \wedge (x_3 \vee \overline{x_5} \vee x_6) \wedge (x_3 \vee \overline{x_6} \vee x_4)$



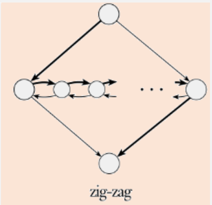


Want: Satisfiable 3cnf formula ⇔ graph with Hamiltonian path

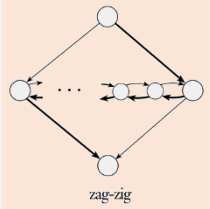
=> If there is Satisfying assignment, then Hamiltonian path exists

These hit all nodes except extra c_js

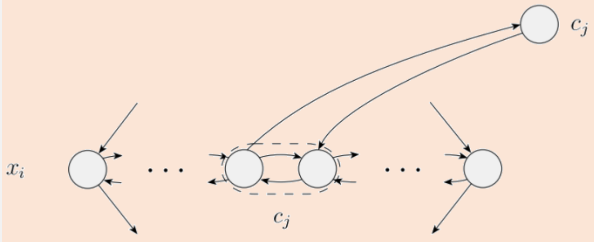
- x_i = TRUE → Hampath "zig-zags" gadget x_i



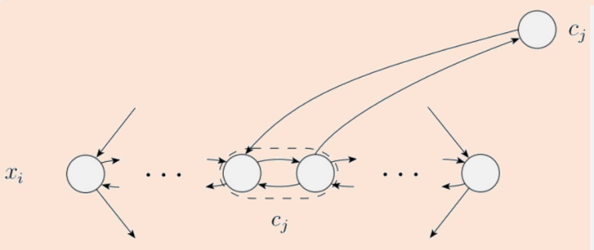
- x_i = FALSE → Hampath "zag-zigs" gadget x_i



- Lit x_i makes clause c_j TRUE → "detour" to c_j in gadget x_i



- Lit $\overline{x_i}$ makes clause c_j TRUE → "detour" to c_j in gadget x_i

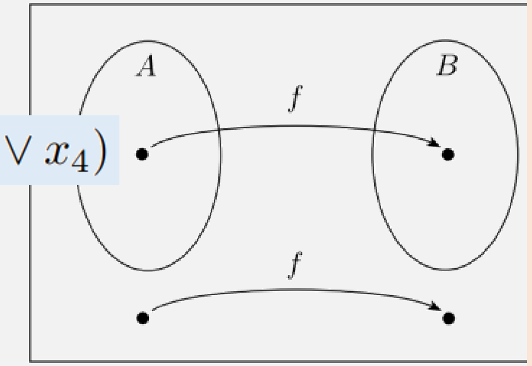


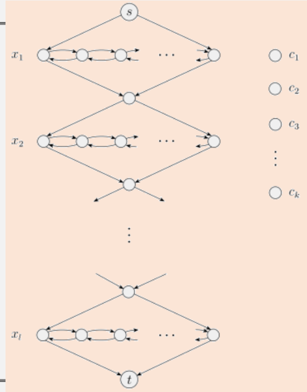
Now path goes through every node

Every clause must be TRUE so path hits all c_j nodes

- And edge directions align with TRUE/FALSE assignments

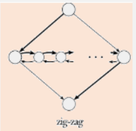
$(x_1 \vee \overline{x_2} \vee \overline{x_3}) \wedge (x_3 \vee \overline{x_5} \vee x_6) \wedge (x_3 \vee \overline{x_6} \vee x_4)$

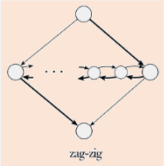




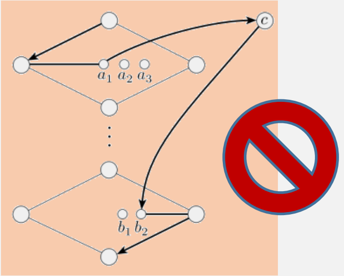
<u>Want</u>: Satisfiable 3cnf formula ⇔ graph with Hamiltonian path

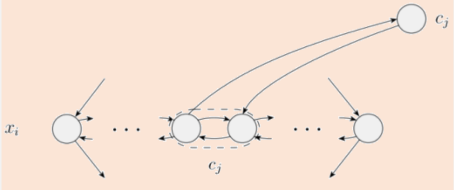
<= if output has Ham. path, then input had Satisfying assignment

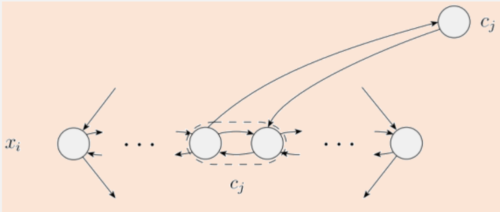
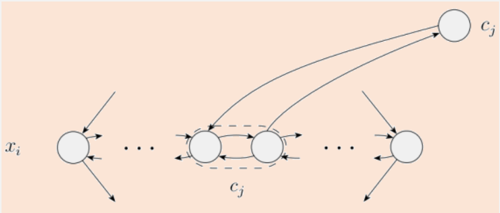




- A Hamiltonian path must choose to either zig-zag or zag-zig gadgets
- Ham path can only hit "detour" c_j nodes by coming right back
- Otherwise, it will miss some nodes





-  gadget x_i "detours" from left to right → x_i = TRUE
-  gadget x_i "detours" from right to left → x_i = FALSE

Theorem: *UHAMPATH* is NP-complete

un

$UHAMPATH = \{\langle G, s, t\rangle |\ G$ is a directed graph with a Hamiltonian path from s to $t\}$

- Reduce *HAMPATH* to *UHAMPATH* (using Thm 7.36)
  - HW11
  - Remember to write out the 5 steps!

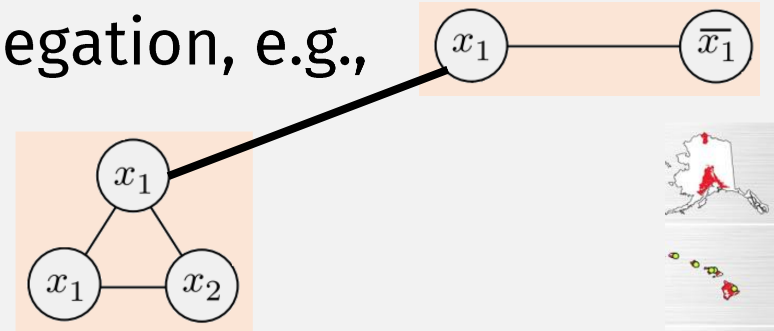
THEOREM 7.36

If B is NP-complete and $B \leq_{\mathrm{P}} C$ for C in NP, then C is NP-complete.

Theorem: *VERTEX-COVER* is NP-complete.

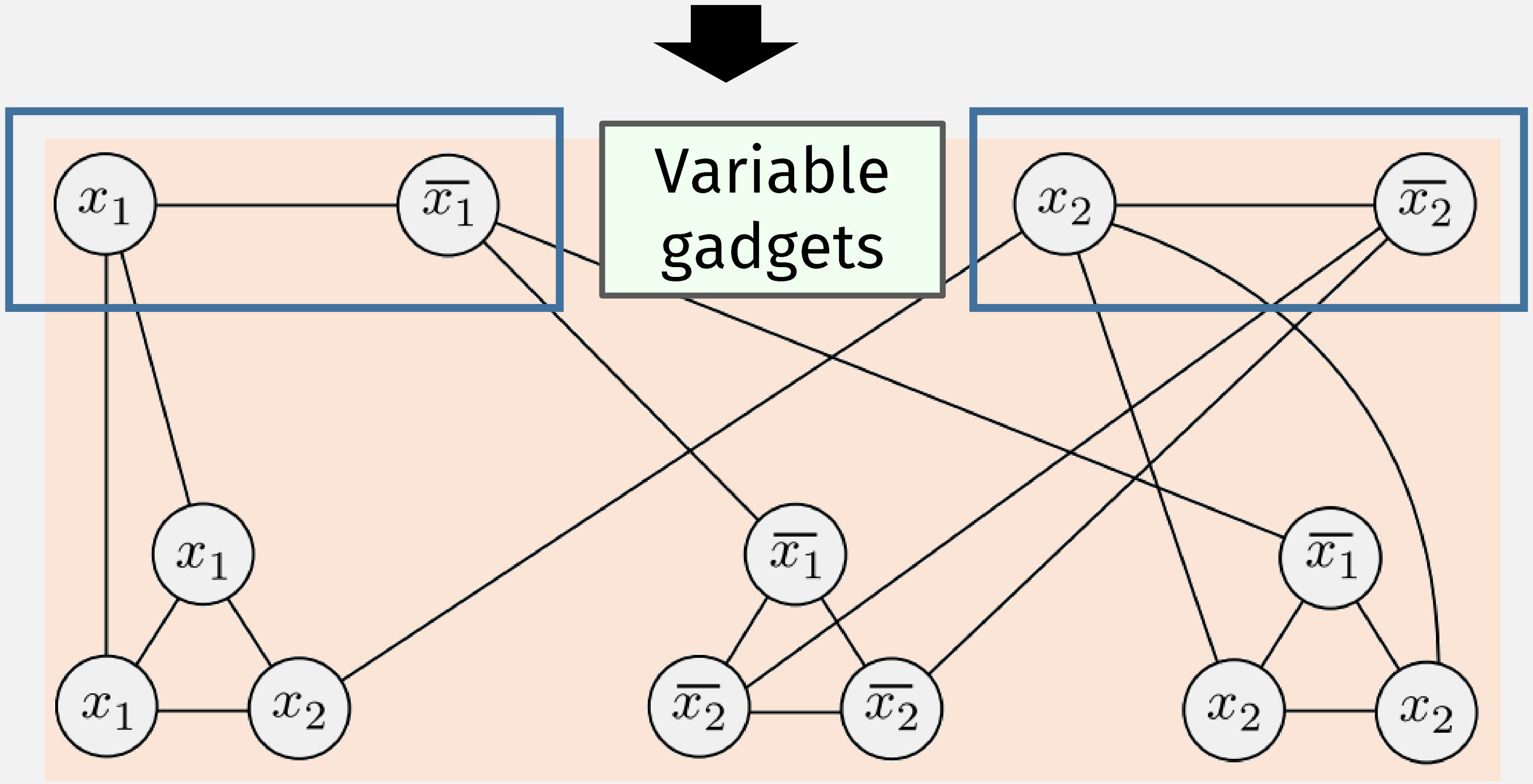
$VERTEX\text{-}COVER = \{\langle G, k\rangle \mid G$ is an undirected graph that has a k-node vertex cover$\}$

- A vertex cover of a graph is …
  - … a subset of its nodes where every edge touches one of those nodes
- Proof Sketch: Reduce *3SAT* to *VERTEX-COVER*
- The reduction maps:
- Variable x_i → 2 connected nodes
  - corresponding to the var and its negation, e.g.,
- Clause → 3 connected nodes
  - corresponding to its literals, e.g.,
- Additionally,
  - connect var and clause gadgets by …
  - … connecting nodes that correspond to the same literal



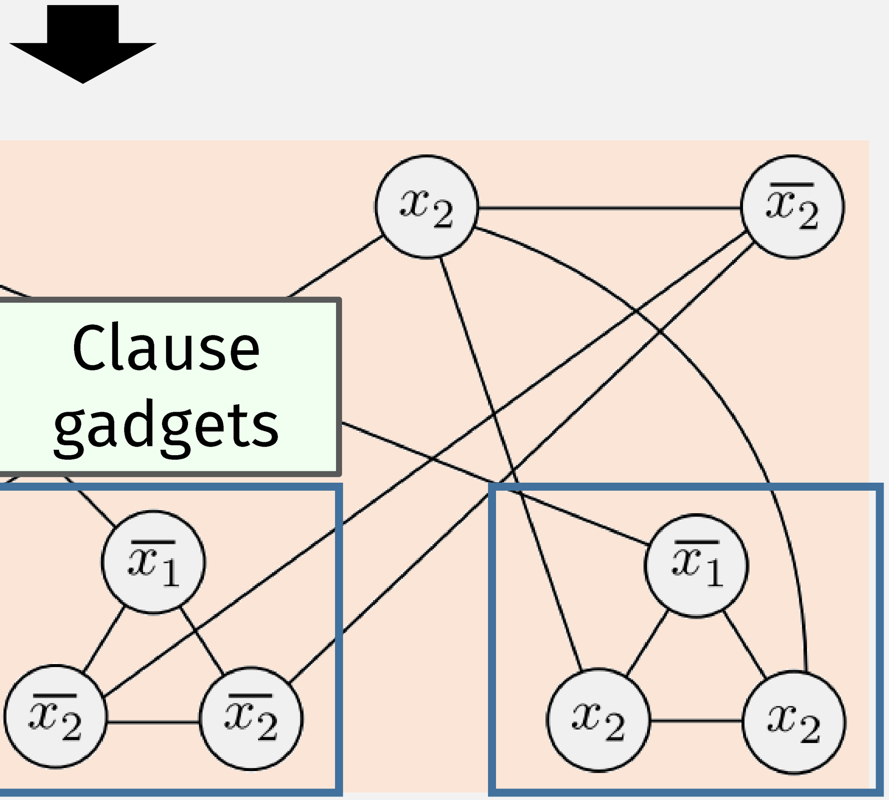
VERTEX-COVER example

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



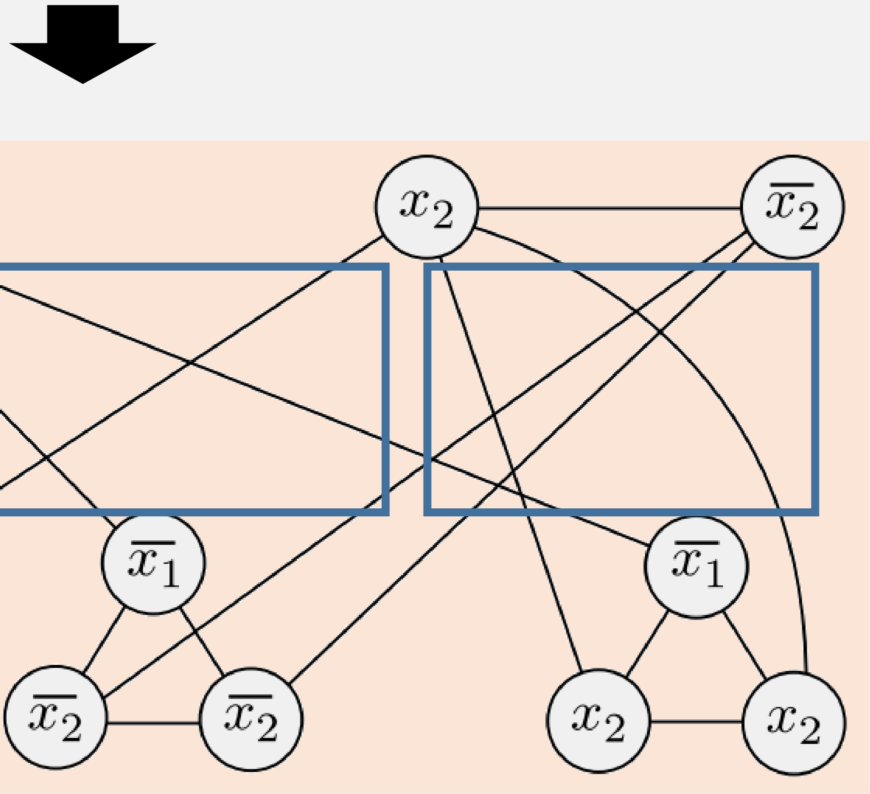
VERTEX-COVER example

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



VERTEX-COVER example

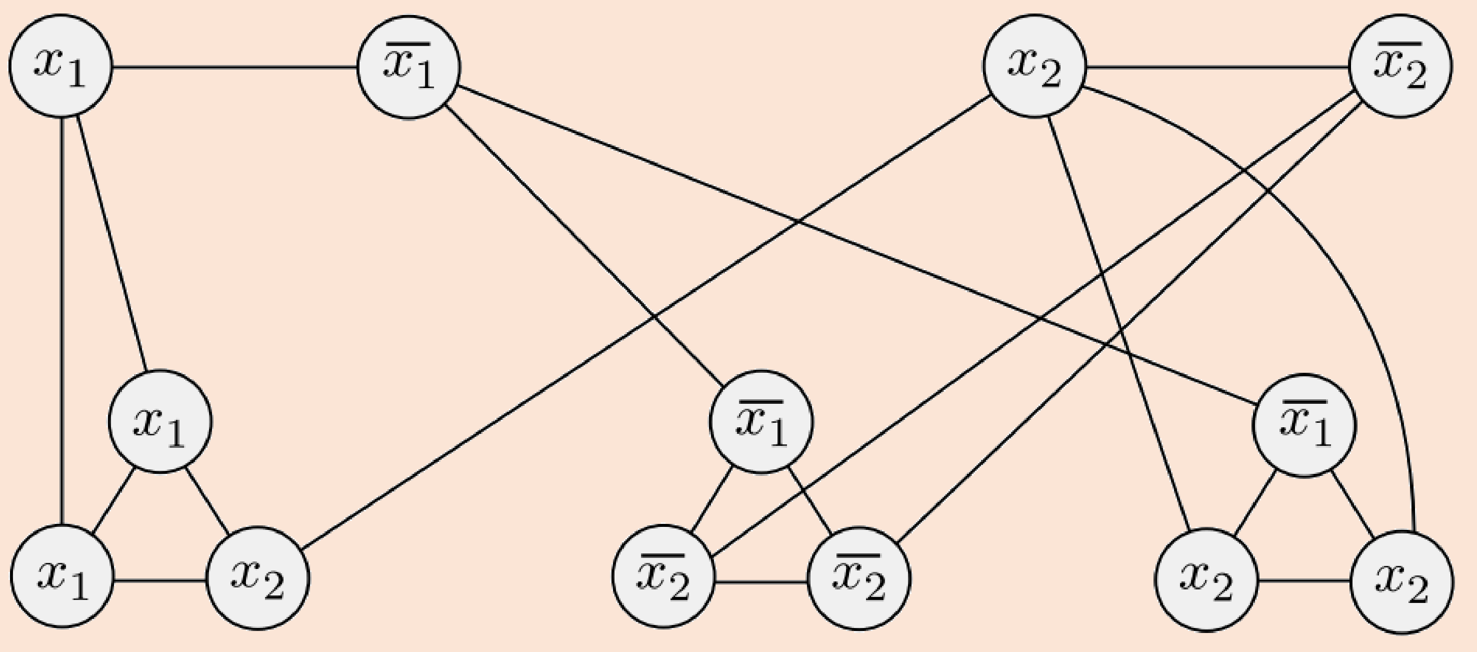
$$\phi = (x_1 \vee x_1 \vee x_2) \ \wedge \ (\overline{x_1} \vee \overline{x_2} \vee \overline{x_2}) \ \wedge \ (\overline{x_1} \vee x_2 \vee x_2)$$



Extra edges connecting variable and clause gadgets together

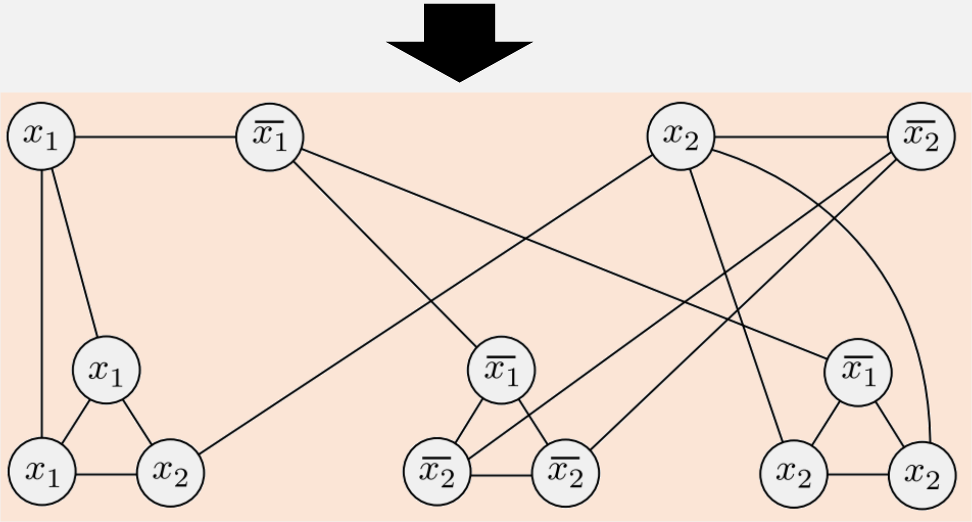
VERTEX-COVER example

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



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

VERTEX-COVER example



- Let formula have …
  - ***m*** = # variables
  - ***l*** = # clauses
- Then graph has …
  - # nodes = 2***m*** + 3***l***

=> If satisfying assignment,

  - then show there is a ***k***-cover where ***k*** = ***m*** + 2***l***
- Nodes in the cover:
  - In each of ***m*** var gadgets, choose 1 node corresponding to TRUE literal
  - For each of ***l*** clause gadgets, ignore 1 TRUE literal and choose other 2
  - Since there is satisfying assignment, each clause has a TRUE literal
  - Total = ***m*** + 2***l***

Take home exercise:
Show that formula is satisfiable
⇔
Graph has a vertex cover with *k* nodes

Check-in Quiz 5/5

On gradescope