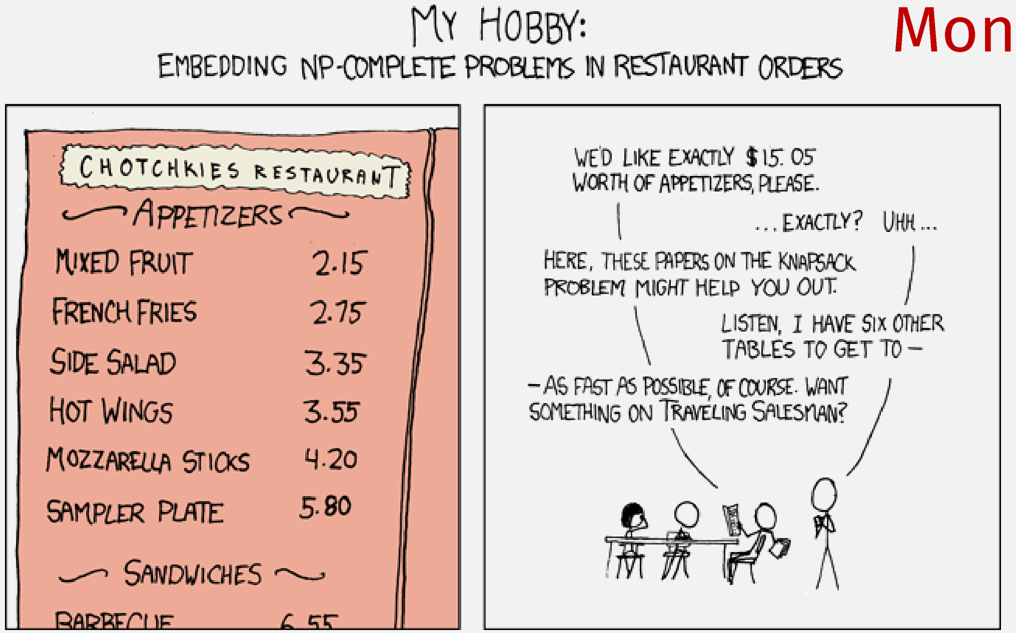
More NP-Complete Problems

Monday, December 7, 2020



HW questions?

Announcements

- HW11 out
    - Last homework

- HW7 grades returned

THEOREM 7.36

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

Proof:

- C is **NP**-complete (Def 7.34) if:
  - 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 → C by:
  - First reduce A → B in poly time
    - Because B is **NP**-Complete
  - Then reduce B → C in poly time
    - This is given
- Total run time: Poly time + poly time = poly time

THEOREM 7.36

known

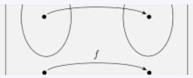
unknown

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 …

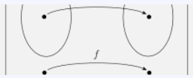
- … by constructing poly time reduction from:
  - $SAT = \{\langle\phi\rangle \mid \phi \text{ is a satisfiable Boolean formula}\}$ (known to be **NP**-Complete)

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

Using: If B is NP-complete and $B \leq_{\mathrm{P}} C$ 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:
  - $SAT = \{\langle\phi\rangle \mid \phi \text{ is a satisfiable Boolean formula}\}$ (known to be **NP**-Complete)

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

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

NP-Complete problems, so far

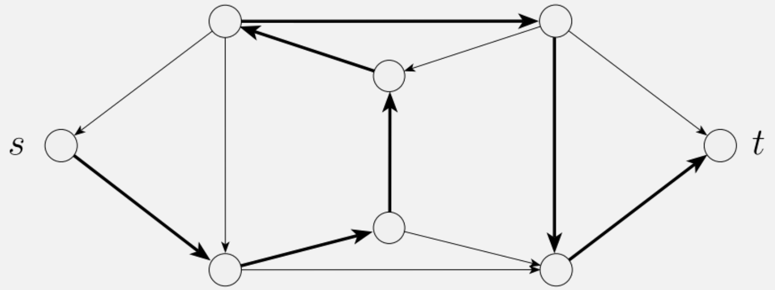
- $SAT = \{\langle\phi\rangle \mid \phi \text{ is a satisfiable Boolean formula}\}$ (Cook-Levin Theorem)
- $3SAT = \{\langle\phi\rangle \mid \phi \text{ is a satisfiable 3cnf-formula}\}$ (reduce *SAT* to *3SAT*)
- $CLIQUE = \{\langle G, k\rangle \mid G \text{ is an undirected graph with a } k\text{-clique}\}$ 
  - *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 $B \leq_{\mathrm{P}} C$ for C in NP, then C is NP-complete.

Other **NP** Problems, so far

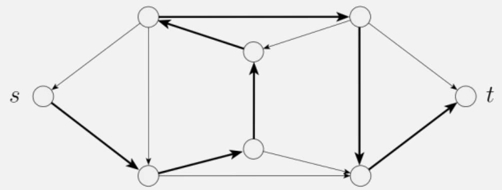
- $HAMPATH = \{\langle G, s, t\rangle \mid G \text{ is a directed graph with a Hamiltonian path from } s \text{ to } t\}$
  - A Hamiltonian path goes through every node in the graph



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



$HAMPATH = \{\langle G, s, t\rangle \mid G$ 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

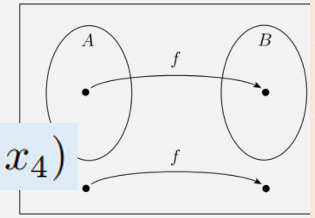
Proof Parts (5):

1. Show *HAMPATH* is in **NP** (done in prev class)
2. Choose **NP**-complete problem to reduce from: *3SAT*
3. Create the computable function:

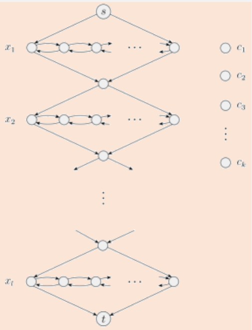
DEFINITION 7.29

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,

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



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



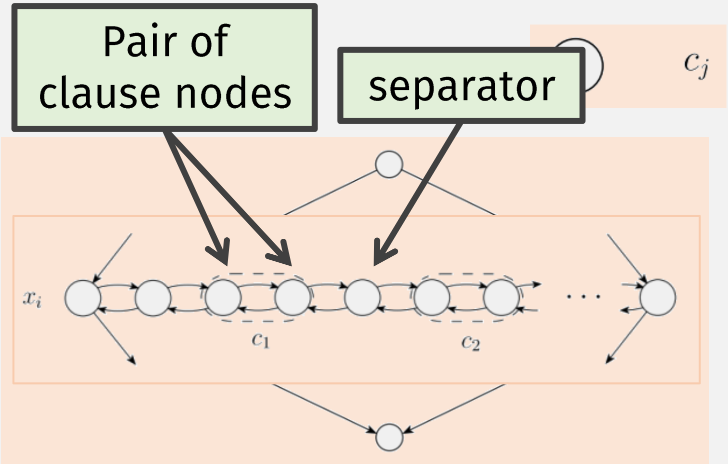
4. Show it runs in poly time
5. Show Def 7.29 iff requirement:
   - Satisfiable 3cnf formula ⇔ graph with Hamiltonian path

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
- Variable → diamond-shaped graph "gadget"
  - Clause → 2 "connector" nodes + separator
  - Total = 3k+1 "connector" nodes per "gadget"

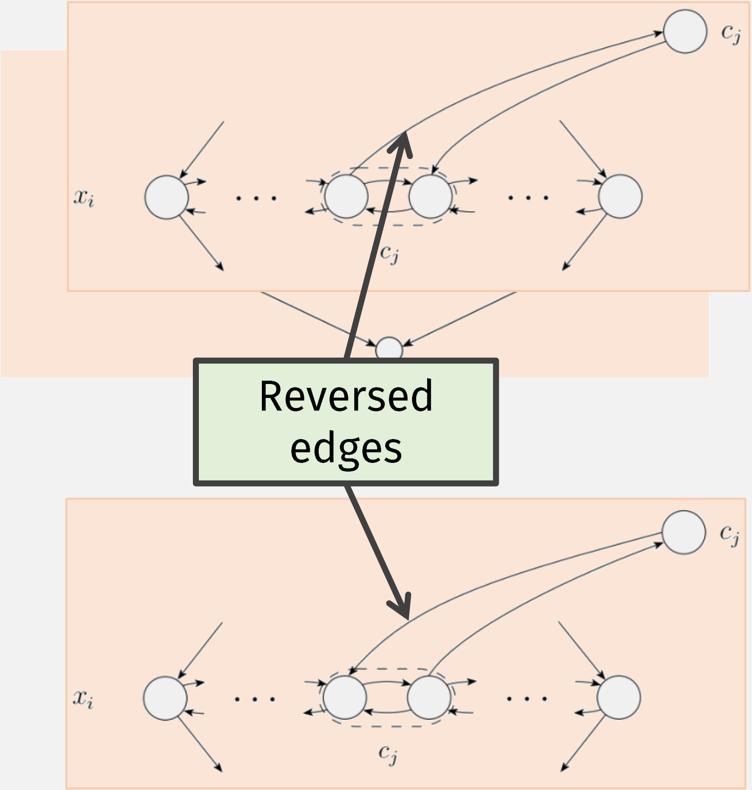


Computable Fn: Formula (blue) → Graph (orange)

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

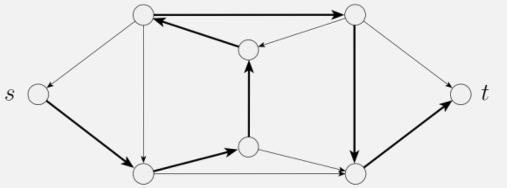
k = # clauses

- Clause → (extra) single nodes
- 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)

<u>Theorem</u>: *HAMPATH* is NP-complete

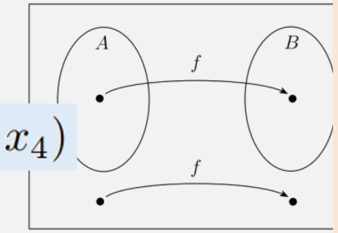


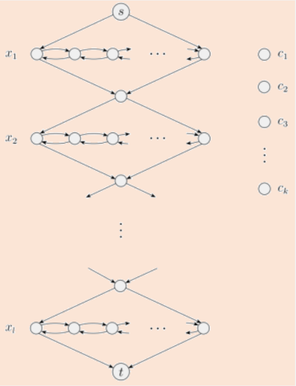
$HAMPATH = \{\langle G, s, t\rangle \mid G \text{ is a directed graph with a Hamiltonian path from } s \text{ to } t\}$

<u>Proof Parts (5)</u>:

1. ~~Show *HAMPATH* is in **NP** (done in prev class)~~
2. ~~Choose **NP**-complete problem to reduce from: *3SAT*~~
3. ~~Create the <u>computable function</u>:~~

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





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

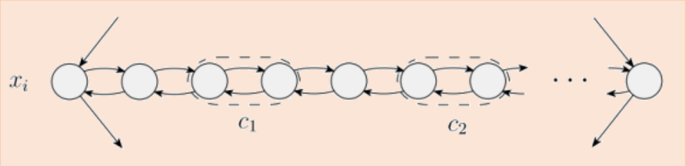
Polynomial Time?

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

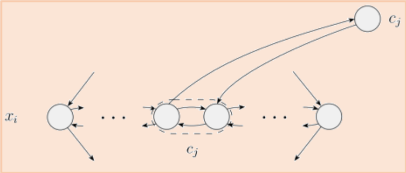
Example input: $\phi = (a_1 \vee b_1 \vee c_1) \wedge (a_2 \vee b_2 \vee c_2) \wedge \cdots \wedge (a_k \vee b_k \vee c_k)$

k = # clauses = at most **3k** variables

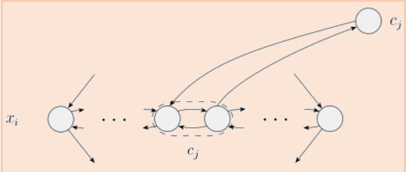
- Clause → (extra) single nodes  $O(\mathbf{k})$
- Variable → diamond-shaped graph "gadget" $O(\mathbf{k}^2)$
  - Clause → 2 "connector" nodes + separator
  - Total = 3k+1 "connector" nodes per "gadget"



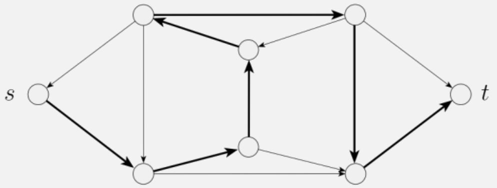
- Lit x_i in clause c_j → c_j node edges in gadget x_i $O(\mathbf{k})$



- Lit $\overline{x}_i$ in clause c_j → c_j edges in gadget x_i (rev) $O(\mathbf{k})$



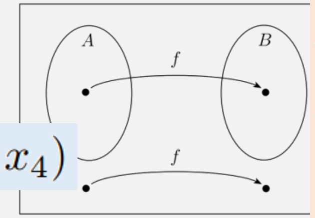
Theorem: *HAMPATH* is NP-complete

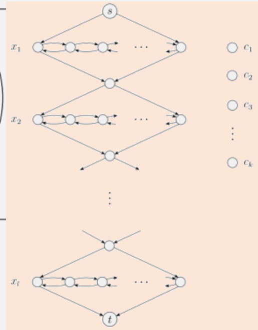


$HAMPATH = \{\langle G, s, t \rangle \mid G$ is a directed graph with a Hamiltonian path from s to $t\}$

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:~~





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

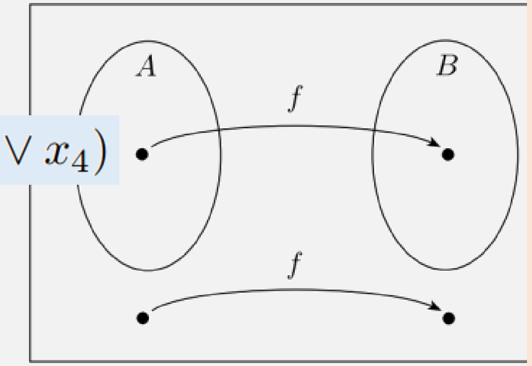
4. ~~Show it runs in poly time~~
5. Show Def 7.29 iff requirement:
   - Satisfiable 3cnf formula ⇔ graph with Hamiltonian path

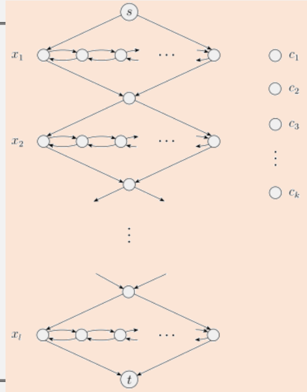
DEFINITION 7.29

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,

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

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



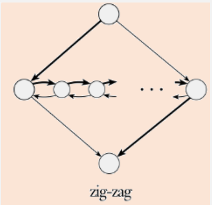


Want: Satisfiable 3cnf formula ⇔ graph with Hamiltonian path

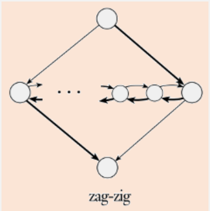
- Satisfying assignment **=>** Hamiltonian path

These hit all nodes except extra c_js

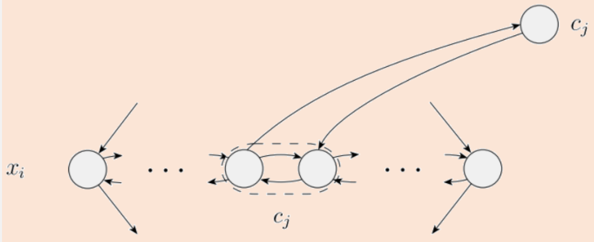
- x_i = TRUE => Hampath "zig-zags" gadget x_i



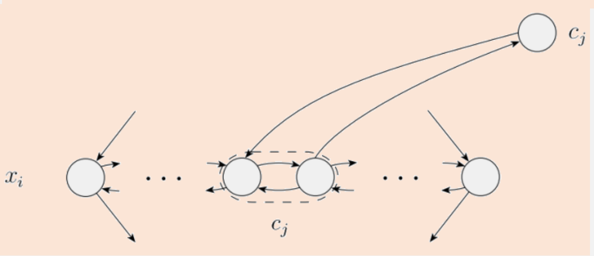
- x_i = FALSE => Hampath "zag-zigs" gadget x_i



- 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

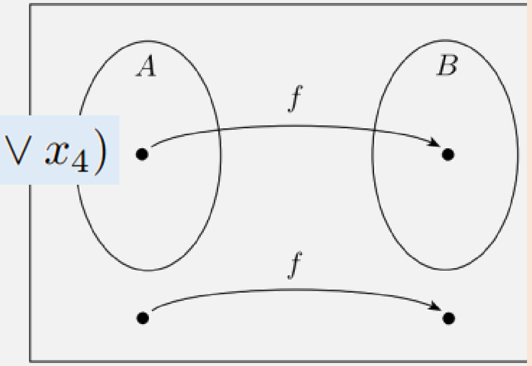


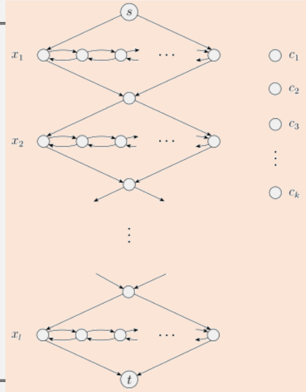
Now path goes through every node

Every clause must be TRUE so path hits all c_j nodes

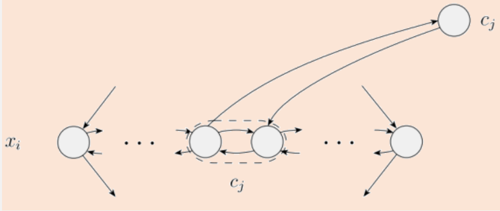
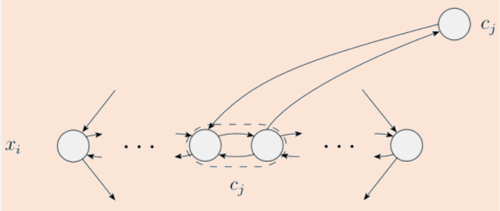
- And edge directions align with TRUE/FALSE assignments

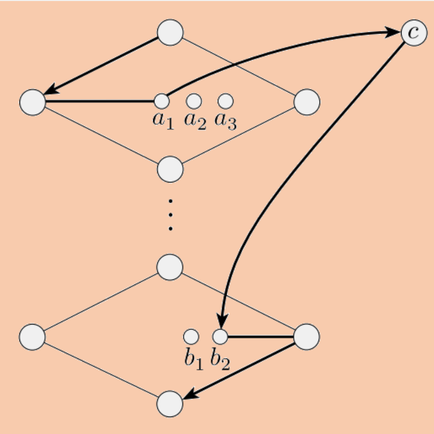
$(x_1 \vee \overline{x_2} \vee \overline{x_3}) \wedge (x_3 \vee \overline{x_5} \vee x_6) \wedge (x_3 \vee \overline{x_6} \vee x_4)$





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

- Hamiltonian path **=>** Satisfying assignment
  -  gadget x_i "detours" from left to right → x_i = TRUE
  -  gadget x_i "detours" from right to left → x_i = FALSE
  - What about "weird" paths?
    - Cannot be Hamiltonian path because it misses some nodes



Theorem: *UHAMPATH* is NP-complete

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

- Reduce *HAMPATH* to *UHAMPATH* (using Thm 7.36)
  - HW11

THEOREM 7.36

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

Theorem: *SUBSET-SUM* is NP-complete

$SUBSET\text{-}SUM = \{\langle S, t\rangle \mid S = \{x_1, \ldots, x_k\}$, and for some $\{y_1, \ldots, y_l\} \subseteq \{x_1, \ldots, x_k\}$, we have $\Sigma y_i = t\}$

THEOREM 7.36

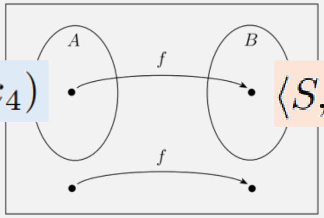
If B is NP-complete and $B \leq_P C$ for C in NP, then C is NP-complete.

Strategy: Use

Proof Parts (5):

1. Show *SUBSET-SUM* is in **NP** (done in prev class)
2. Choose **NP**-complete problem to reduce from: *3SAT*
3. Create the computable function *f*:

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



$\langle S, t\rangle \mid S = \{x_1, \ldots, x_k\}$

4. Show it runs in poly time
5. Show Def 7.29 iff requirement:

ϕ is a satisfiable 3cnf-formula $\iff$ $f(\langle \phi \rangle) = \langle S, t\rangle$ where some subset of S sums to t

Computable Fn: 3cnf → $\langle S, t \rangle$

E.g., $(x_1 \vee \overline{x_2} \vee x_3) \wedge (x_2 \vee x_3 \vee \cdots) \wedge \cdots \wedge (\overline{x_3} \vee \cdots \vee \cdots)$ ➡

- Assume formula has:
  - ***l*** variables $x_1, \ldots, x_l$
  - ***k*** clauses $c_1, \ldots, c_k$
- Computable function *f* maps:
  - Variable x_i → two numbers y_i and z_i
  - Clause c_j → two numbers g_j and h_j
- Each number has max ***l+k*** digits:
- Sum is ***l*** 1s followed by ***k*** 3s

y_i and z_i: i^{th} digit = 1

y_i: ***l***+j^{th} digit = 1 if c_j has x_i

z_i: ***l***+j^{th} digit = 1 if c_j has $\overline{x_i}$

g_j and h_j: ***l***+j^{th} digit = 1

The sum

	1	2	3	4	$\cdots$	l	c_1	c_2	$\cdots$	c_k
y_1	1	0	0	0	$\cdots$	0	1	0	$\cdots$	0
z_1	1	0	0	0	$\cdots$	0	0	0	$\cdots$	0
y_2		1	0	0	$\cdots$	0	0	1	$\cdots$	0
z_2		1	0	0	$\cdots$	0	1	0	$\cdots$	0
y_3			1	0	$\cdots$	0	1	1	$\cdots$	0
z_3			1	0	$\cdots$	0	0	0	$\cdots$	1
$\vdots$					$\ddots$	$\vdots$	$\vdots$		$\vdots$	$\vdots$
y_l						1	0	0	$\cdots$	0
z_l						1	0	0	$\cdots$	0
g_1							1	0	$\cdots$	0
h_1							1	0	$\cdots$	0
g_2								1	$\cdots$	0
h_2								1	$\cdots$	0
$\vdots$									$\ddots$	$\vdots$
g_k										1
h_k										1
t	1	1	1	1	$\cdots$	1	3	3	$\cdots$	3

Theorem: *SUBSET-SUM* is NP-complete

SUBSET-SUM $= \{\langle S, t\rangle \mid S = \{x_1, \ldots, x_k\}$, and for some $\{y_1, \ldots, y_l\} \subseteq \{x_1, \ldots, x_k\}$, we have $\Sigma y_i = t\}$

Strategy: Use

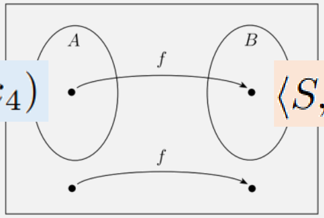
THEOREM 7.36

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

Proof Parts (5):

1. Show ~~*SUBSET-SUM* is in **NP** (done in prev class)~~
2. ~~Choose **NP**-complete problem to reduce from: *3SAT*~~
3. ~~Create the computable function *f*:~~

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



$\langle S, t\rangle \mid S = \{x_1, \ldots, x_k\}$

4. Show it runs in poly time
5. Show Def 7.29 iff requirement:

ϕ is a satisfiable 3cnf-formula $\iff$ $f(\langle\phi\rangle) = \langle S, t\rangle$ where some subset of S sums to t

Polynomial Time?

E.g., $(x_1 \vee \overline{x_2} \vee x_3) \wedge (x_2 \vee x_3 \vee \cdots) \wedge \cdots \wedge (\overline{x_3} \vee \cdots \vee \cdots)$ ➡

- Assume formula has:
  - ***l*** variables $x_1, \ldots, x_l$
  - ***k*** clauses $c_1, \ldots, c_k$
- Table size: (***l*** + ***k***)(2***l*** + 2***k***)
  - Creating it requires at most a constant number of passes over the table
  - Num variables ***l*** = 3***k*** at most
- Total: O(***k*** ²)

	1	2	3	4	$\cdots$	l	c_1	c_2	$\cdots$	c_k
y_1	1	0	0	0	$\cdots$	0	1	0	$\cdots$	0
z_1	1	0	0	0	$\cdots$	0	0	0	$\cdots$	0
y_2		1	0	0	$\cdots$	0	0	1	$\cdots$	0
z_2		1	0	0	$\cdots$	0	1	0	$\cdots$	0
y_3			1	0	$\cdots$	0	1	1	$\cdots$	0
z_3			1	0	$\cdots$	0	0	0	$\cdots$	1
$\vdots$					$\ddots$	$\vdots$	$\vdots$		$\vdots$	$\vdots$
y_l						1	0	0	$\cdots$	0
z_l						1	0	0	$\cdots$	0
g_1							1	0	$\cdots$	0
h_1							1	0	$\cdots$	0
g_2								1	$\cdots$	0
h_2								1	$\cdots$	0
$\vdots$									$\ddots$	$\vdots$
g_k										1
h_k										1
t	1	1	1	1	$\cdots$	1	3	3	$\cdots$	3

Theorem: *SUBSET-SUM* is NP-complete

$SUBSET\text{-}SUM = \{\langle S,t\rangle \mid S = \{x_1,\ldots,x_k\},$ and for some $\{y_1,\ldots,y_l\} \subseteq \{x_1,\ldots,x_k\},$ we have $\Sigma y_i = t\}$

Strategy: Use

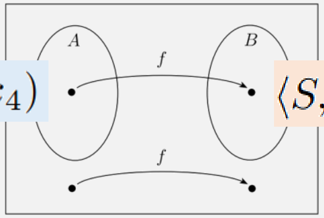
THEOREM 7.36

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

Proof Parts (5):

1. Show ~~*SUBSET-SUM* is in **NP** (done in prev class)~~
2. ~~Choose **NP**-complete problem to reduce from: *3SAT*~~
3. ~~Create the computable function *f*:~~

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



$\langle S,t\rangle \mid S = \{x_1,\ldots,x_k\}$

4. ~~Show it runs in poly time~~
5. Show Def 7.29 iff requirement:

ϕ is a satisfiable 3cnf-formula $\iff$ $f(\langle\phi\rangle) = \langle S,t\rangle$ where some subset of S sums to t

ϕ is a satisfiable 3cnf-formula $\iff$ $f(\langle\phi\rangle) = \langle S, t\rangle$ where some subset of

Each column:
- At least one 1
- At most 3 1s

- => If formula is satisfiable, choose …
  - Sum t = l 1s followed by k 3s
  - S to include:
    - y_i if x_i = TRUE
    - z_i if x_i = FALSE
    - and some of g_i and h_i to make the sum t
- Numbers in S sum to t because:
  - Left digits:
    - only one of y_i or z_i is in S
  - Right digits:
    - Top: Each column sums to 1, 2, or 3
      - Because each clause has only 3 literals
    - Bottom:
      - Add g_i and/or h_i to make column sum to 3

S only includes one

	1	2	3	4	$\cdots$	l	c_1	c_2	$\cdots$	c_k
y_1	1	0	0	0	$\cdots$	0	1	0	$\cdots$	0
z_1	1	0	0	0	$\cdots$	0	0	0	$\cdots$	0
y_2		1	0	0	$\cdots$	0	0	1	$\cdots$	0
z_2		1	0	0	$\cdots$	0	1	0	$\cdots$	0
y_3			1	0	$\cdots$	0	1	1	$\cdots$	0
z_3			1	0	$\cdots$	0	0	0	$\cdots$	1
$\vdots$					$\ddots$	$\vdots$	$\vdots$		$\vdots$	$\vdots$
y_l						1	0	0	$\cdots$	0
z_l						1	0	0	$\cdots$	0
g_1							1	0	$\cdots$	0
h_1							1	0	$\cdots$	0
g_2								1	$\cdots$	0
h_2								1	$\cdots$	0
$\vdots$									$\ddots$	$\vdots$
g_k										1
h_k										1
t	1	1	1	1	$\cdots$	1	3	3	$\cdots$	3

Determines if x_i or $\overline{x_i}$ is in clause c_j

ϕ is a satisfiable 3cnf-formula $\iff$ $f(\langle\phi\rangle) = \langle S, t\rangle$ where some subset of S sums to t

- <= If f creates S with numbers summing to t
  - Formula has l variables, k clauses, and has …
  - lit $\overline{x_i}$ in clause c_j if i^{th} number pair (1st) has l+j^{th} digit = 1
  - lit x_i in clause c_j if i^{th} number pair (2nd) has l+j^{th} digit = 1
- There must be a satisfying assignment:
  - x_i = TRUE if y_i in S
  - x_i = FALSE if z_i in S
- This is satisfying because:
  - For each column c_j
    - g_j and h_j total at most 2
    - so at least 1 number from top is included satisfy sum t
    - Which means at least one literal in every clause makes it makes it TRUE

	1	2	3	4	$\cdots$	l	c_1	c_2	$\cdots$	c_k
y_1	1	0	0	0	$\cdots$	0	1	0	$\cdots$	0
z_1	1	0	0	0	$\cdots$	0	0	0	$\cdots$	0
y_2		1	0	0	$\cdots$	0	0	1	$\cdots$	0
z_2		1	0	0	$\cdots$	0	1	0	$\cdots$	0
y_3			1	0	$\cdots$	0	1	1	$\cdots$	0
z_3			1	0	$\cdots$	0	0	0	$\cdots$	1
$\vdots$					$\ddots$	$\vdots$	$\vdots$		$\vdots$	$\vdots$
y_l						1	0	0	$\cdots$	0
z_l						1	0	0	$\cdots$	0
g_1							1	0	$\cdots$	0
							1	0	$\cdots$	0
								1	$\cdots$	0
								1	$\cdots$	0
$\vdots$									$\ddots$	$\vdots$
g_k										1
h_k										1
t	1	1	1	1	$\cdots$	1	3	3	$\cdots$	3

In each column, accounts for at most 2 out of 3 required sum of 3

Check-in Quiz 12/7

On gradescope

End of Class Survey 12/7

See course website