Objectives

- Proving NP-complete by reduction
- Example NP-complete languages cover:
  - 3SAT
  - CLIQUE
  - INDEPENDENT SET
  - VERTEX COVER

Conjuctive Normal Form

- A literal is a Boolean variable or a negated Boolean variable. E.g., x, $\neg y$
- A clause is several literals connected with $\vee$'s. E.g., $(x \vee y \vee \neg z)$
- A Boolean formula is in Conjuctive Normal Form (Don't confuse this with Chomosky Normal Form!!!) if it is made of clauses connected with $\wedge$'s.

  E.g., $(x \vee y \vee \neg z) \wedge (\neg y \vee z) \wedge (\neg x)$

CNF-SAT is NP-complete

A Boolean formula is a cnf-formula if it is a formula in Conjuctive Normal Form

Let CNF-SAT be the language

$$\{ \langle F \rangle \mid F \text{ is a satisfiable cnf-formula } \}$$

Theorem: CNF-SAT is NP-complete.

CNF-SAT is NP-complete (2)

Proof: To show CNF-SAT is NP-complete, we notice that:

- CNF-SAT is in NP (easy to prove)
- Every language in NP is polynomial time reducible to CNF-SAT
  - Because the proof of Cook-Levin theorem in Lecture 20 can be directly re-used (recall that the reduction is based on cnf-formula)

Thus, CNF-SAT is NP-complete

3SAT is NP-complete

A Boolean formula is a 3cnf-formula if it is a formula in Conjuctive Normal Form, and every clause has exactly 3 literals

Let 3SAT be the language

$\{ \langle F \rangle \mid F$ is a satisfiable 3cnf-formula $\}$

Theorem: 3SAT is NP-complete.

3SAT is NP-complete (2)

Proof: To show 3SAT is NP-complete, two things to be done:

- Show 3SAT is in NP (easy)
- Show that every language in NP is polynomial time reducible to 3SAT (how?)

  ➔ Sufficient to give polynomial time reduction from some NP-complete language to 3SAT (why?)

Which NP-complete language shall we use?

3SAT is NP-complete (3)

To reduce CNF-SAT to 3SAT, we convert a cnf-formula F into a 3cnf-formula F', with

F is satisfiable ⇔ F' is satisfiable

Firstly, let $C_1, C_2, \ldots, C_k$ be the clauses in F.
If F is a 3cnf-formula, just set F' to be F.
Otherwise, the only reasons why F is not a 3cnf-formula are:

- Some clauses C_i has less than 3 literals
- Some clauses C_i has more than 3 literals

3SAT is NP-complete (4)

- For each clause that has one literal, say L_1, we change it into $(L_1 \vee L_1 \vee L_1)$
  - ➔ Thus, if F' is satisfiable, the value of L_1 must be 1
- For each clause that has two literals, say $(L_1 \vee L_2)$, we change it into $(L_1 \vee L_2 \vee L_1)$
  - ➔ Thus, if F' is satisfiable, the value of $(L_1 \vee L_2)$ must be 1

3SAT is NP-complete (5)

- For each clause that has more than three literals, say $(L_1 \vee L_2 \vee \dots \vee L_m)$, we use new variables z_i, and replace the clause by

$$(L_1 \vee L_2 \vee z_1) \wedge (\neg z_1 \vee L_3 \vee z_2) \wedge$$
$$(\neg z_2 \vee L_4 \vee z_3) \wedge \dots \wedge (\neg z_{m-3} \vee L_{m-1} \vee L_m)$$

➔ Thus, if F' is satisfiable, the value of $(L_1 \vee L_2 \vee \dots \vee L_m)$ must be 1 (why??)

3SAT is NP-complete (6)

- Finally, for each clause that has three literals, no change to it

By our construction of F',

F is satisfiable ⇔ F' is satisfiable (why??)

Also, the above conversion takes polynomial time (why??) So, CNF-SAT is polynomial time reducible to 3SAT

Thus, 3SAT is NP-complete

CLIQUE is NP-complete

Recall that CLIQUE is the language

$\{ \langle G, k \rangle \mid G$ is a graph with a k-clique $\}$

Theorem: CLIQUE is NP-complete.

How to prove??

CLIQUE is NP-complete (2)

Proof: To show CLIQUE is NP-complete, two things to be done:

- Show CLIQUE is in NP (done before)
- Show that every language in NP is polynomial time reducible to CLIQUE

→ Sufficient to give polynomial time reduction from some NP-complete language to CLIQUE

Which NP-complete language shall we use?

CLIQUE is NP-complete (3)

Let us try to reduce 3SAT to CLIQUE:

Let F be a 3cnf-formula.
Let $C_1, C_2, \ldots, C_k$ be the clauses in F.
Let $x_{j,1}$, $x_{j,2}$, $x_{j,3}$ be the literals of C_j.

Hint: Construct a graph G such that

F is satisfiable $\Leftrightarrow$ G has a k-clique

CLIQUE is NP-complete (4)

Proof (cont.):

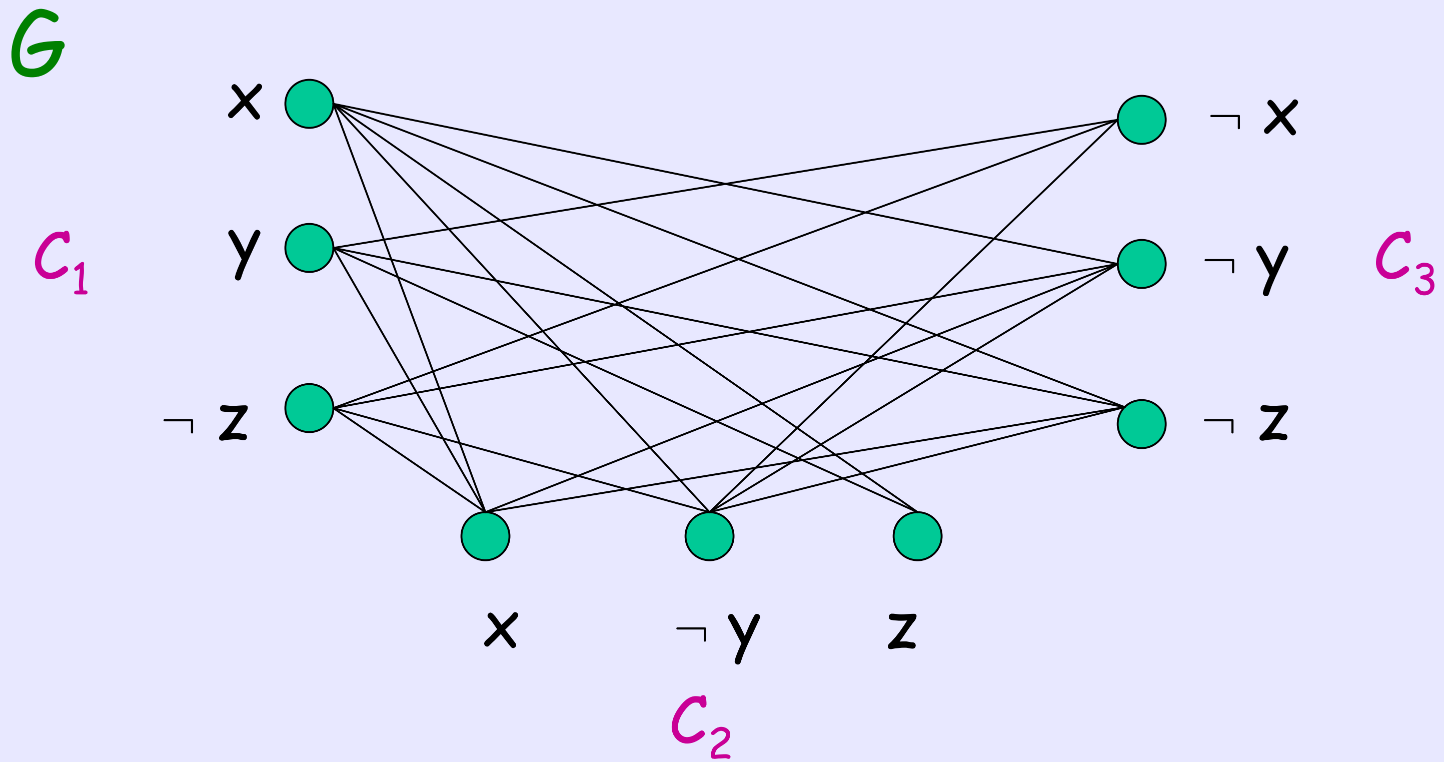
We construct a graph G as follows:

1. For each literal $x_{j,q}$, we create a distinct vertex in G representing it
2. G contains all edges, except those
   (i) joining two vertices in same clause,
   (ii) joining two vertices whose literals is the negation of the others

E.g., (see next slide)

Constructing G from F

$F = (x \vee y \vee \neg z) \wedge (x \vee \neg y \vee z) \wedge$
$(\neg x \vee \neg y \vee \neg z)$



CLIQUE is NP-complete (5)

Proof (cont.): We now show that

G has a k-clique ⇔ F is satisfiable

(=>) If G has a k-clique,

1. the k-clique must a vertex from each clause (why?)
2. also, no vertex will be the negation of the others in the clique (why?)

Thus, by setting the corresponding literal (not variable) to TRUE, F will be satisfied

CLIQUE is NP-complete (6)

(<=) If F is satisfiable, at least a literal in each clause is set to TRUE in the satisfying assignment

So, the corresponding vertices forms a clique (why?) Thus, G has a k-clique

Finally, since G can be constructed from F in polynomial time, so we have a polynomial time reduction from 3SAT to CLIQUE

Thus, CLIQUE is NP-complete

IND-SET is NP-complete

A set of vertices inside a graph G is an independent set if there are no edges between any two of these vertices

Let IND-SET be the language

$\{ \langle G,k \rangle \mid G$ is a graph with an independent set of size $k \}$

Theorem: IND-SET is NP-complete.

IND-SET is NP-complete (2)

Proof: To show IND-SET is NP-complete, two things to be done:

- Show IND-SET is in NP (easy)
- Show every language in NP is polynomial time reducible to IND-SET

→ Sufficient to give polynomial time reduction from some NP-complete language to IND-SET

Hint: Use CLIQUE for the reduction

IND-SET is NP-complete (3)

Proof (cont.):

We shall construct G' such that

G has a k-clique

$\Leftrightarrow$

G' has an independent set of size k

That is, construct G' such that

$\langle G,k \rangle$ in CLIQUE $\Leftrightarrow$ $\langle G',k \rangle$ in IND-SET

IND-SET is NP-complete (4)

Given $G=(V,E)$, we set $G'=(V',E')$ to be the complement of G. In other words, $V = V'$ (G and G' has the same set of vertices), but e in $E \Leftrightarrow e$ not in E'

It is easy to check that G' is the desired graph we want (why?). As the construction of G' is done in polynomial time, CLIQUE is polynomial time reducible to IND-SET

Thus, IND-SET is NP-complete.

VERTEX-COVER is NP-complete

A set of vertices inside a graph G is a **vertex cover** if every edge in G is connected to at least one vertex in the set.

Let VERTEX-COVER be the language

$\{ \langle G,k \rangle \mid G$ is a graph with a vertex cover of size $k \}$

Theorem: VERTEX-COVER is NP-complete.

VERTEX-COVER is NP-complete (2)

Proof: To show VERTEX-COVER is NP-complete, two things to be done:

- Show VERTEX-COVER is in NP (easy)
- Show that every language in NP is poly-time reducible to VERTEX-COVER
  - ➔ Sufficient to give polytime reduction from a NP-complete language to VERTEX-COVER

Hint: Use IND-SET for the reduction

VERTEX-COVER is NP-complete (3)

Proof (cont.):

Let $G=(V,E)$ with n vertices.
We will show that

$$G \text{ has an independent set of size } k$$
$$\Leftrightarrow$$
$$G \text{ has a vertex cover of size } n-k$$

That is, we show

$$\langle G,k\rangle \text{ in IND-SET} \Leftrightarrow \langle G,n-k\rangle \text{ in VERTEX-COVER}$$

VERTEX-COVER is NP-complete (4)

If V' is a vertex cover of G, every edge of G is attached to at least one vertex in V'. So, if we delete V', no edge remains.

Thus, V-V' will be an independent set.

On the other hand, if V-V' is an independent set, V' must be a vertex cover (why?).

So, IND-SET is polynomial time reducible to VERTEX-COVER (how is the reduction done??)

Thus, VERTEX-COVER is NP-complete

Next Time

- More NP-complete problems