Large-scale Knowledge Processing

Lecture 4

Kazuhisa Seto

Today's Lecture

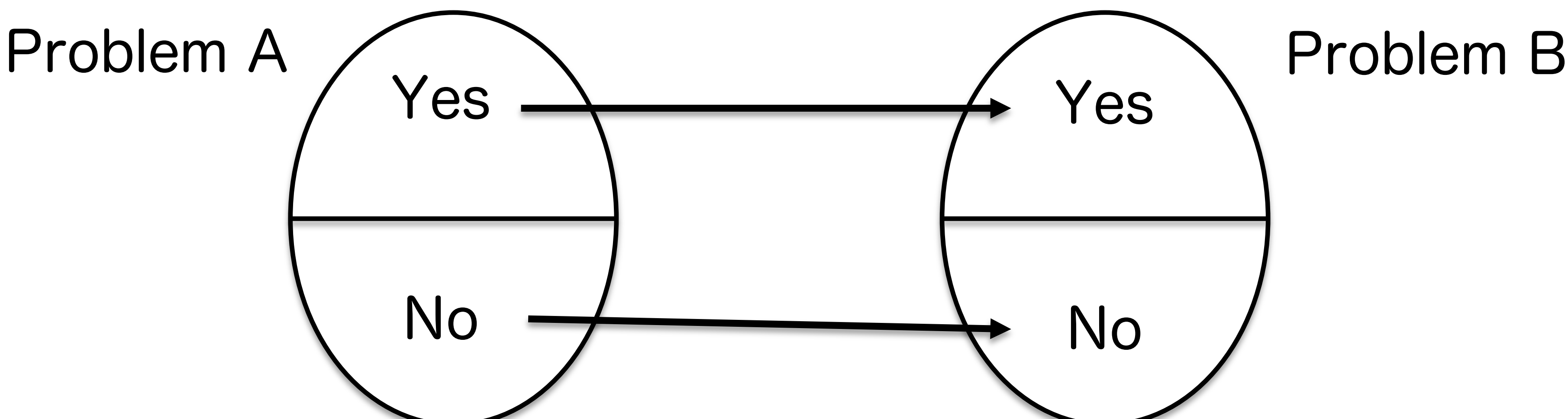
Study the foundation on theory of computation

- Reduction
- NP-complete Problems and Polynomial-time Reducitons
- Polynomial-time Reduction from CNF-SAT to 3SAT
- Polynomial-time Reduction from 3SAT to Vertex Cover Problem

Reduction

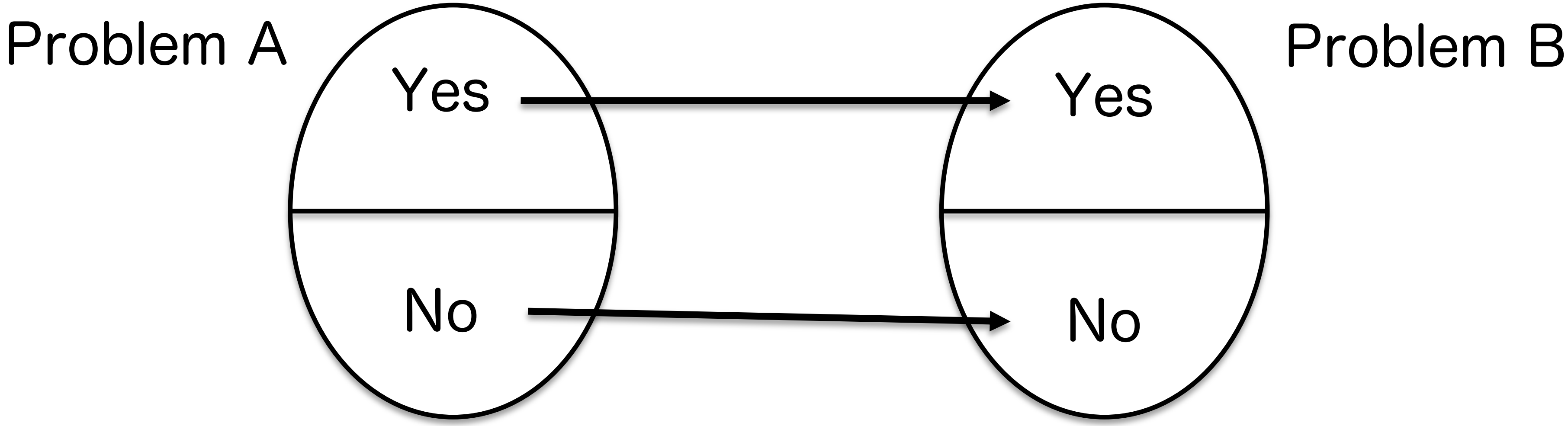
The definition of reduction

A reduction is a map from an instance of problem A to an instance of problem B such that an instance of problem A outputs Yes if and only if a mapping instance of problem B also outputs Yes.



Reduction is useful

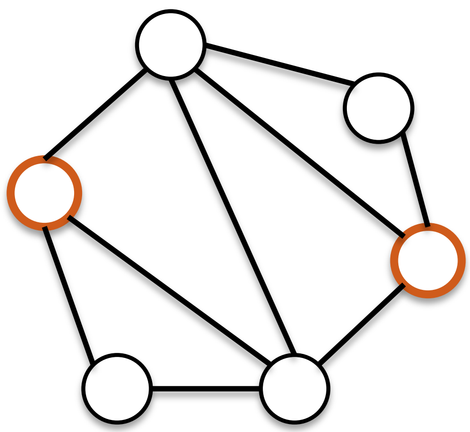
When one wants to solve a decision problem A, by using reduction, we can get the answer of A by solving the problem B.



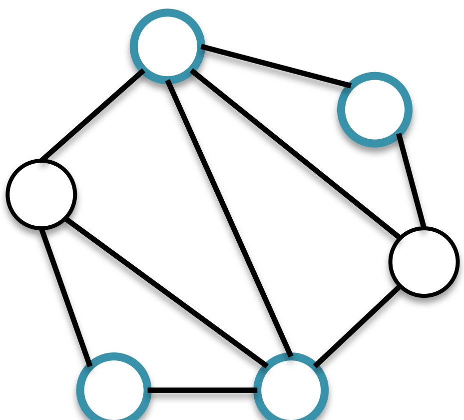
An example of reduction

We consider the following two problems.

- Independent Set Problem : IS
- Vertex Cover Problem : VC



Independent Set



Vertex Cover

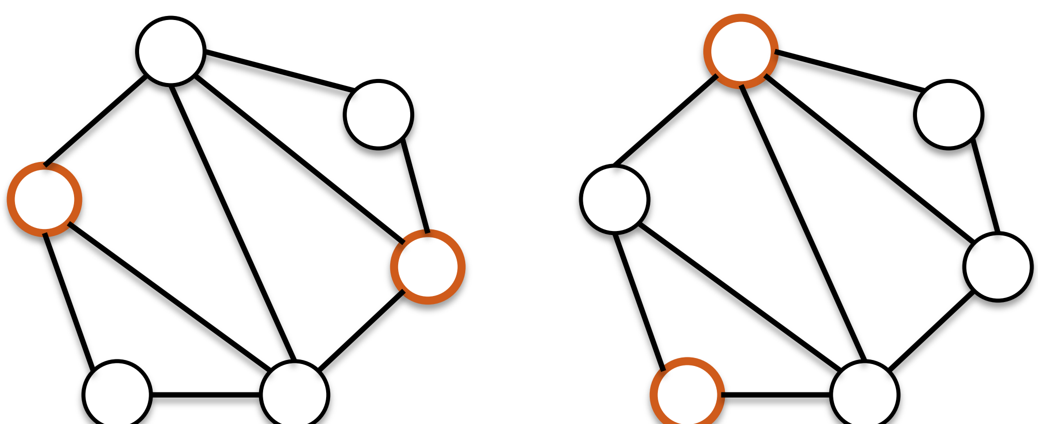
Independent Set Problem (IS)

Input：A Graph G and a positive integer k

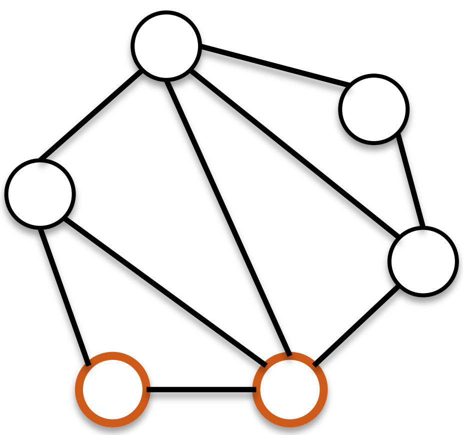
Ask：Is there an independent set of size k in G

Independent Set：A set I such that any two elements in I are not adjacent in G.

Note that the empty set ∅ is an independent set.



Orange circles are independent set



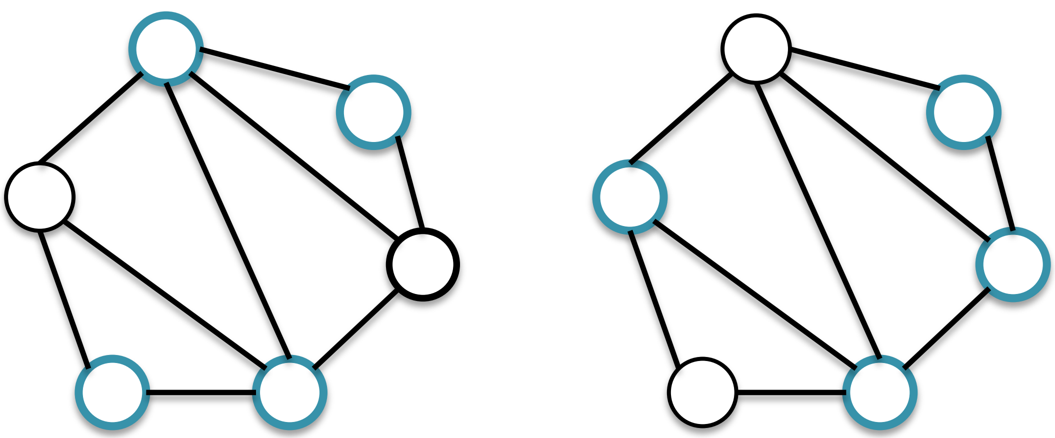
Not independent set

Vertex Cover (VC)

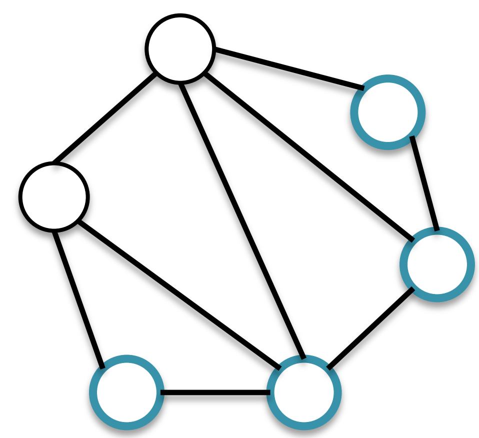
Input : A graph G and a positive integer k

Ask : Is there a vertex cover of size k in G ?

Vertex Cover : A set of vertices C such that for every edge e, at least one endpoint of e in C.



Blue circles are vertex cover



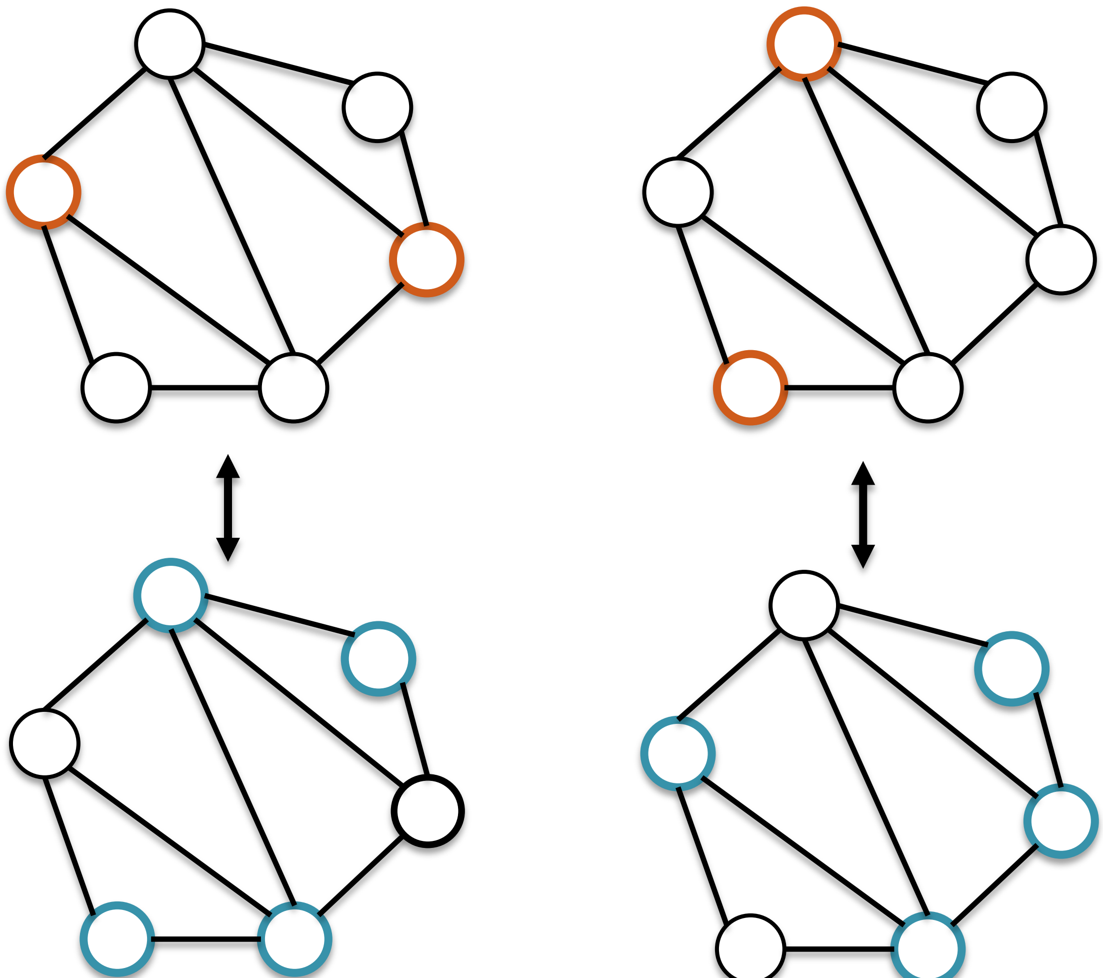
Not vertex cover

The relation between IS and VC

Orange Circles (IS) + Blue Circles (VC) = All Vertices

IS (k=2)

VC (k=4)



The relation between IS and VC

There exists an independent set of size k in G

⇕

There exists a vertex cover of size n-k in G

Let V be a vertex set of G. Let S be an independent set in G and T be a vertex set V-S

⇒ Since S is an IS, then there is no edge between any pair of vertices in S.

⇒ Every vertex in S has an edge to some vertex in T.

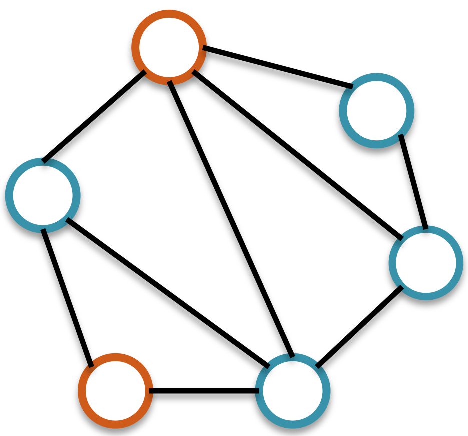
⇒ T must be a vertex cover. (See a figure in next slide)

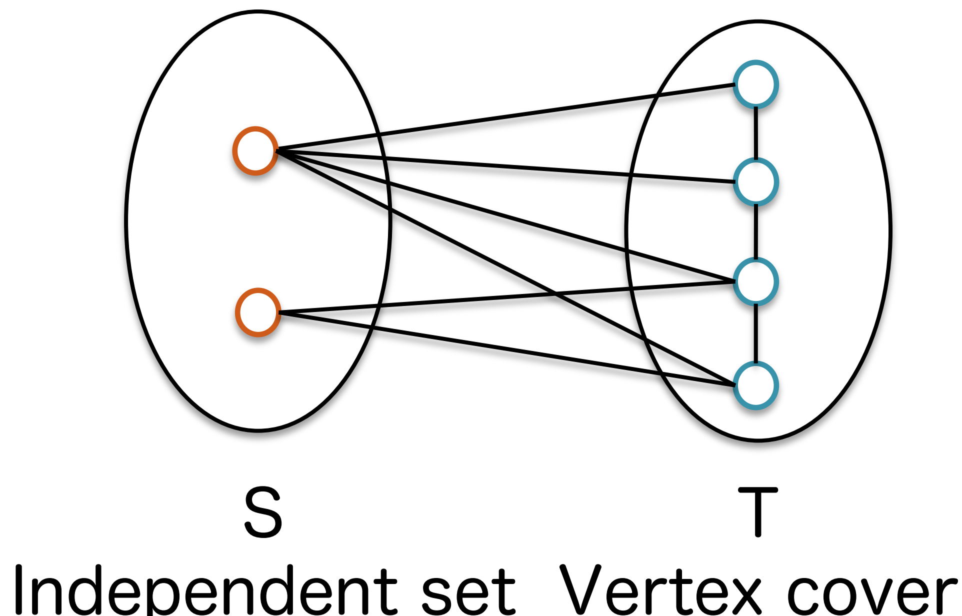
The relation between IS and VC

There exists an independent set of size k in G

$\Updownarrow$

There exists a vertex cover of size n-k in G





A reduction from IS to VC

Input：A Graph G and a positive integer k

Ask (IS)：Is there an independent set of size k in G

Ask (VC)：Is there a vertex cover of size $n - k$ in G ?

If IS outputs Yes、then VC outputs Yes.

⇒ Does it hold that if IS outputs No, then VC outputs No ?

We consider a contraposition.

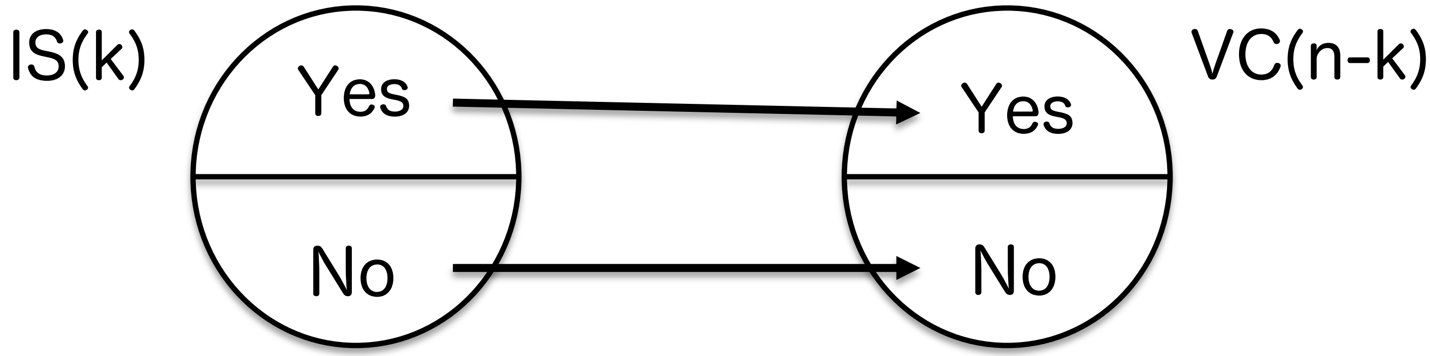
Does is hold that if VC outputs Yes, then IS outputs Yes ?

⇒ It holds. （Consider why it holds by yourself)

A reduction from IS to VC

There exists the following map from an instance of IS to an instance of VC. This is a reduction.

- If there exists an independent set of size k in G, then there exist a vertex cover of size n-k in G.
- If there exists no independent set of size k in G, then there exists no vertex cover of size n-k in G.



NP-complete problems and polyonomial-time reductions

Class P and Class NP

Class P（Polynomial time）

- A set of decision problems solved by deterministic TMs in polynomial time.

Class NP（Nondeterministic Polynomial time）

- A set of decision problems solved by non-deterministic TMs in polynomial time.
- A set of decision problems such that when a given x and a witness w, deterministic TMs decide that x is yes instances in polynomial time.

NP-complete problem (NPC)

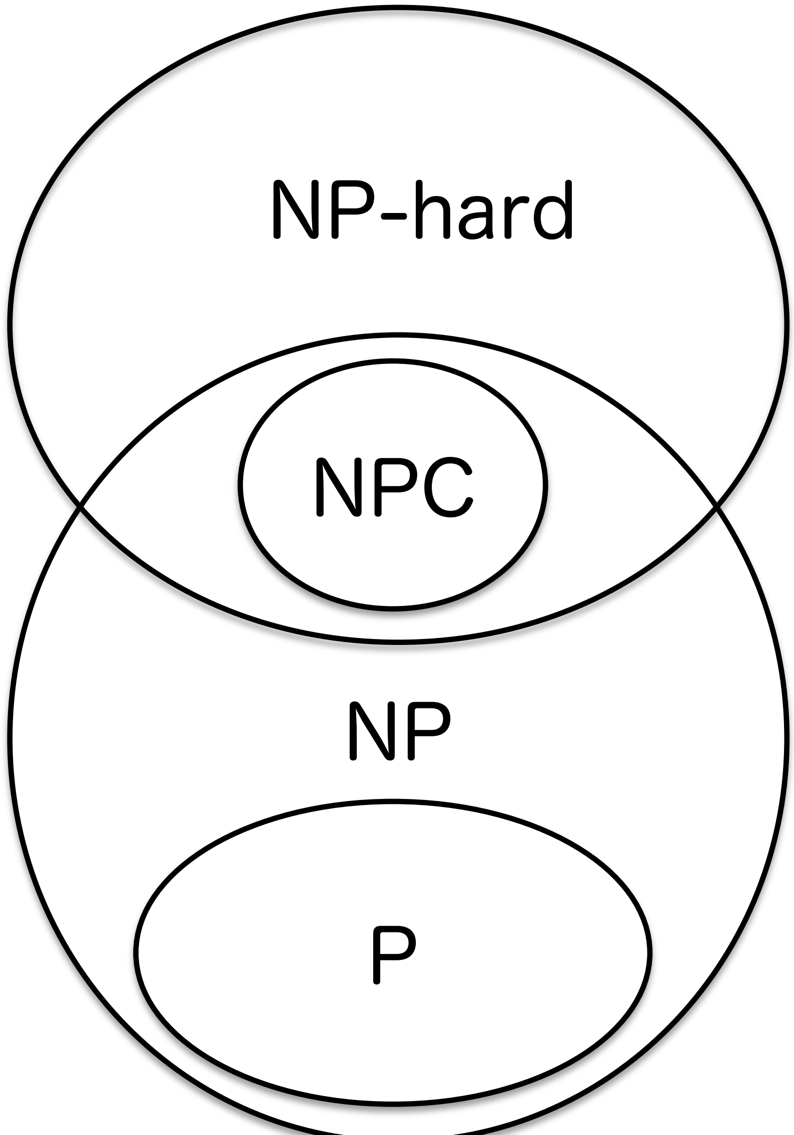
The definition of NP-complete Problem

A decision problem L is NP-complete if

- L is in NP.
- Any problem in NP can be reducible to L in deterministic polynomial time. (NP-hard)

Polynomial-time Reduction: A reduction from an instance of problem A to an instance of problem B by determinist TMs in polynomial time .

The inclusion relation between P, NP, NPC and NP-hard



NP-Complete Problem

Intuitively, NPC is the most difficult problems in NP.

⇒ If a problem in NPC can be solved, then all problems in NP can be solved.

⇒ If a NP-complete problem can be in deterministic polynomial time, then P = NP.

The following problems are typical NP-complete problems.

- Satisfiability Problem, Independent Set Problem, Vertex Cover Problem, Hamiltonian Cycle Problem, etc…

Satisfiability Problem（SAT）

Input：Boolean formula ϕ

Ask：Is there exists an assignment to the input variables such as ϕ=1 ?

Satisfying assignment α : an assignment α such as ϕ=1

We say α satisfies ϕ when α is a satisfying assignment.

If ϕ has some satisfying assignment, we say ϕ is satisfiable.

CNF-SAT: The satisfiability problem for CNF.

CNF：$(x_1 \vee \bar{x}_2)(x_1 \vee x_2 \vee x_3)(\bar{x}_1 \vee x_2 \vee x_3)(\bar{x}_1 \vee \bar{x}_3)$

SAT is NP-complete

SAT is the first problem shown that it is NP-complete.

Cook-Levin's Theorem [Cook '71, Levin '73]

SAT（CNF-SAT）is NP-complete.

If a problem has a reduction from SAT, it is also NP-complete problem.

From the next slide, We see some reductions from SAT.

Polynomial Reduction from CNF-SAT to 3SAT

3Satisfiability（3SAT）

<u>kSAT</u>

Input：kCNF ϕ

Ask：Is there exists an assignment to the input variables such as ϕ=1 ?

kCNF：CNF and each clause has at most k literals

3CNF：$(x_1 \vee \bar{x}_2)(x_1 \vee x_2 \vee x_3)(\bar{x}_1 \vee x_2 \vee x_3)(\bar{x}_1 \vee \bar{x}_3)$

3SAT when k=3.

A reduction from 4SAT to 3SAT

First, we show a reduction from 4SAT to 3SAT.

We consider the following clause with 4 literals.

$$C = x_1 \vee \bar{x}_2 \vee \bar{x}_3 \vee x_4$$

We introduce a new variable z to construct two new clauses C_1 and C_2 .

$$C_1 = x_1 \vee \bar{x}_2 \vee z \qquad C_2 = \bar{x}_3 \vee x_4 \vee \bar{z}$$

A reduction from 4SAT to 3SAT

Next, we show the following.

- If C is satisfiable by some assignment α, then α satisfies both C_1 and C_2.

$C = x_1 \vee \bar{x}_2 \vee \bar{x}_3 \vee x_4$ $\quad C_1 = x_1 \vee \bar{x}_2 \vee z$ $\quad C_2 = \bar{x}_3 \vee x_4 \vee \bar{z}$

If α set $x_1 = 1$ or $\bar{x}_2 = 1$, then by setting $z = 0$, both C_1 and C_2 are satisfiable.

If α set $\bar{x}_3 = 1$ or $x_4 = 1$, then by setting $z = 1$, both C_1 and C_2 are satisfiable

Thus, we can replace C by C_1 and C_2.

A reduction from 4SAT to 3SAT

For all clauses with 4 literals, we do the similar replacements, we can convert 4CNF ϕ to 3CNF ϕ' .
If ϕ is satisfiable, then ϕ' is also satisfiable.

But, is it ture that if ϕ is not satisfiable, then ϕ' is also not satisfiable.

- We consider the contraposition.

「If ϕ' is satisfiable、ϕ is also satisfiable.」

A reduction from 4SAT to 3SAT

See the previous example. We show the following.

If both C_1 and C_2 are satisfiable, then C is also satifiable.

$C_1 = x_1 \lor \bar{x}_2 \lor z$ $\quad$ $C_2 = \bar{x}_3 \lor x_4 \lor \bar{z}$ $\quad$ $C = x_1 \lor \bar{x}_2 \lor \bar{x}_3 \lor x_4$

If we set $z = 0$, we must set $x_1 = 1$ or $\bar{x}_2 = 1$ to satisfy C_1.

If we set $z = 1$, we must set $\overline{x_3} = 1$ or $x_4 = 1$ to satisfy C_2.

Bothe case, C is satisfiable.

This holds all clauses, thus if ϕ is not satisfiable, then ϕ' is also not satisfiable.

This is a polynomial-time reduction

This reduction is done in polynomial time.

For an input 4CNF ϕ, n denotes the number of variables in ϕ and m denotes the number of clauses in ϕ .

- One replacement is done in a constant time (=O(1)) because the number of each clause is 4.
- The number of clauses is m, thus the total time of replacements is done in O(m) time.

Input size is n+m, thus this reduction is done in polynomial time because O(m) is polynomial of n+m.

The number of variables (clauses) in ϕ' is at most n+m (2m)

A reduction from kSAT to 3SAT

Almost same as the reduction from 4SAT to 3SAT.

We consider the following clause with k literals.

$$C = x_1 \vee x_2 \vee x_3 \vee x_4 \vee \cdots \vee x_{k-1} \vee x_k$$

We introduce a new variable $z_1, z_2, \ldots, z_{k-3}$ to replace C by the following C'.

$$C' = (x_1 \vee x_2 \vee z_1)(\bar{z}_1 \vee x_3 \vee z_2)(\bar{z}_2 \vee x_4 \vee z_3) \cdots (\bar{z}_{k-3} \vee x_{k-1} \vee x_k)$$

By this replacement, kSAT is reducible to 3SAT.

This is a polynomial-time reduction

This reduction is done in polynomial time.

For an input kCNF ϕ, n denotes the number of variables in ϕ and m denotes the number of clauses in ϕ .

- One replacement is done in O(k) time because the number of each clause is k.
- The number of clauses is m, thus the total time of replacements is done in O(km) time.

Input size is n+m, thus this reduction is done in polynomial time because O(km) is polynomial of n+m.

The number of variables (clauses) in ϕ' is at most n+m(k-3) ((k-2)m).

A reduction from CNF-SAT to 3SAT

CNF-SAT is also kSAT, but k may be n.

However, a reduction is the same as that of kSAT from 3SAT.

Thus, time complexity of this reduction is O(nm) because the previous reduction takes O(km) time and k=n.

It is polynomial time because O(mn) is polynomial of n+m.

Because CNF-SAT is NP-complete and the reduction is polynomial time, thus 3SAT is also NP-complete.

Exercise 1

Reduce the following instance of 4SAT to an instance of 3SAT by the reduction explained in the previous slides.

$$\phi = (x_1 \vee \bar{x}_2 \vee x_3 \vee x_4)(\bar{x}_1 \vee x_2 \vee \bar{x}_3 \vee x_4)(\bar{x}_1 \vee \bar{x}_2 \vee x_3 \vee \bar{x}_4)$$

Exercise 1 (Answer)

At first, for each clause, we divide two clauses by one new variable.

$$\phi = (x_1 \vee \bar{x}_2 \vee x_3 \vee x_4)(\bar{x}_1 \vee x_2 \vee \bar{x}_3 \vee x_4\ )(\bar{x}_1 \vee \bar{x}_2 \vee x_3 \vee \bar{x}_4)$$

$$C_1 = (x_1 \vee \bar{x}_2 \vee x_3 \vee x_4) \rightarrow (x_1 \vee \bar{x}_2 \vee u)(x_3 \vee x_4 \vee \bar{u})$$

$$C_2 = (\bar{x}_1 \vee x_2 \vee \bar{x}_3\ \vee x_4) \rightarrow (\bar{x}_1 \vee x_2 \vee v)(\bar{x}_3\ \vee x_4 \vee \bar{v})$$

$$C_3 = (\bar{x}_1 \vee \bar{x}_2 \vee x_3 \vee \bar{x}_4) \rightarrow (\bar{x}_1 \vee \bar{x}_2 \vee z)(x_3 \vee \bar{x}_4 \vee \bar{z})$$

We combine these clauses with AND and replace u, v, z by z_1, z_2, z_3, respectively.

$$\phi' = (x_1 \vee \bar{x}_2 \vee z_1)(\bar{z}_1 \vee x_3 \vee x_4)(\bar{x}_1 \vee x_2 \vee z_2)(\bar{x}_3 \vee x_4 \vee \bar{z}_2\ )$$

$$\cdot (\bar{x}_1 \vee \bar{x}_2 \vee z_3)(x_3 \vee \bar{x}_4 \vee \bar{z}_3)$$

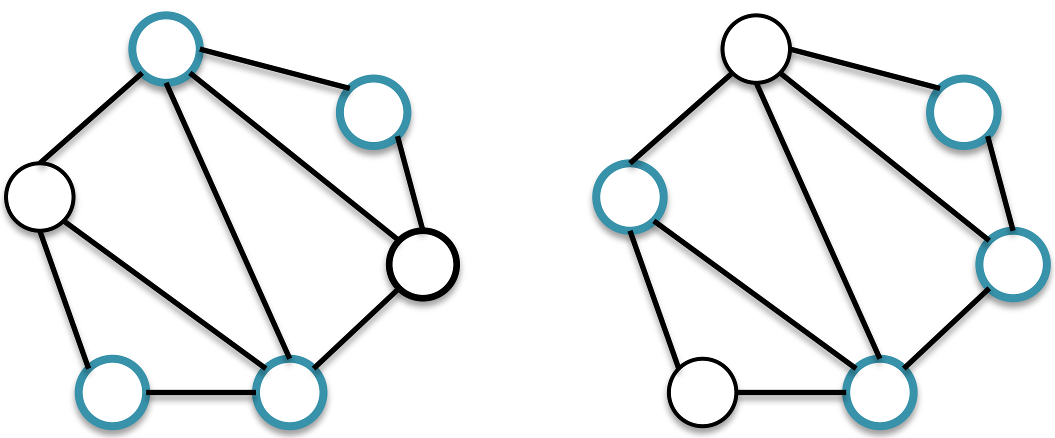
Polynomial-time Reduction from 3SAT to Vertex Cover

Review: Vertex Cover (VC)

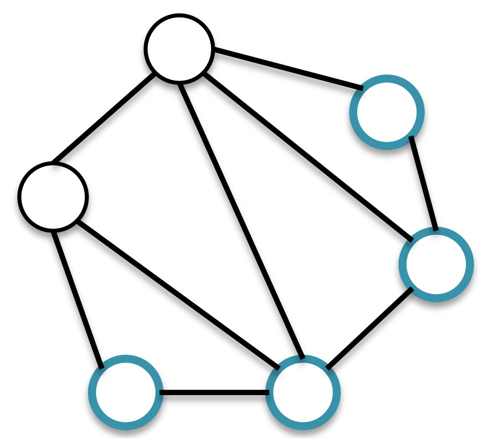
Input : A graph G and a positive integer k

Ask : Is there a vertex cover of size k in G ?

Vertex Cover : A set of vertices C such that for every edge e, at least one endpoint of e in C.



Blue circles are vertex cover



Not vertex cover

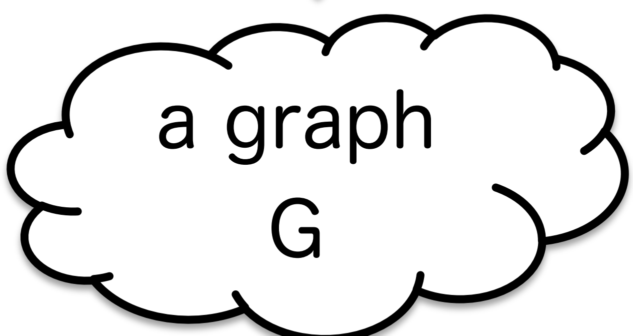
A reduction from 3SAT to VC

A reduction from CNF-SAT to 3SAT is a reduction from a Boolean formula to a Boolean formula.

This reduction is from a Boolean formula to a graph.

For simplicity, all clauses of 3CNF have exact 3 literals.

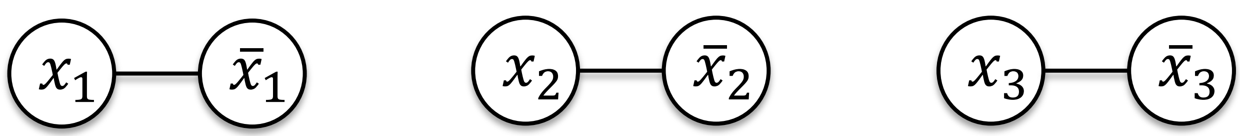
$$\phi = (x_1 \vee \bar{x}_2 \vee x_3)(\bar{x}_1 \vee x_2 \vee \bar{x}_3)(\bar{x}_1 \vee \bar{x}_2 \vee x_3)$$



A reduction from 3SAT to VC

First, we create two gadgets from a given formula ϕ.

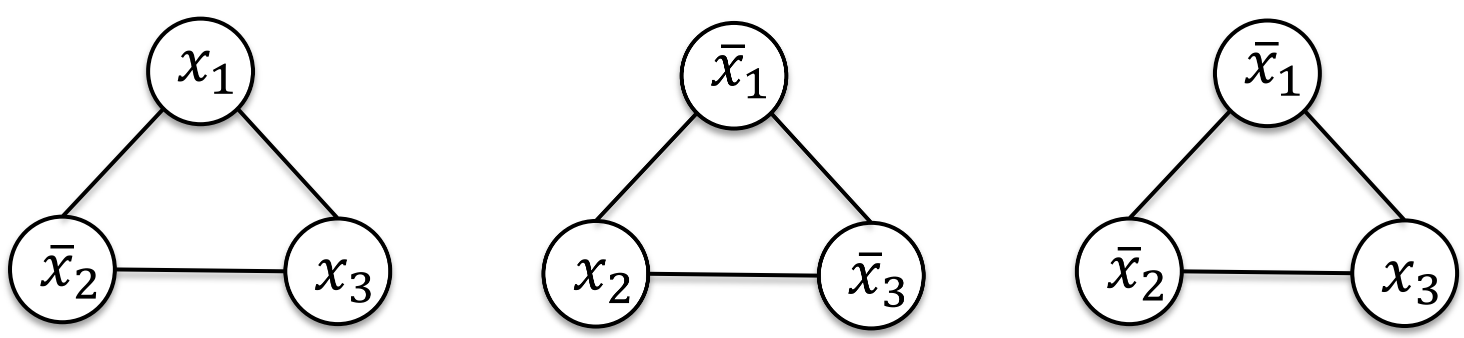
- Variable Gadget
  - ✓ For each variable x_i of ϕ, we create two vertices, one corresponds positive literal x_i and the other corresponds negative literal $\bar{x}_i$. Then, connect these vertices.



A reduction from 3SAT to VC

First, we create two gadgets from a given formula ϕ.

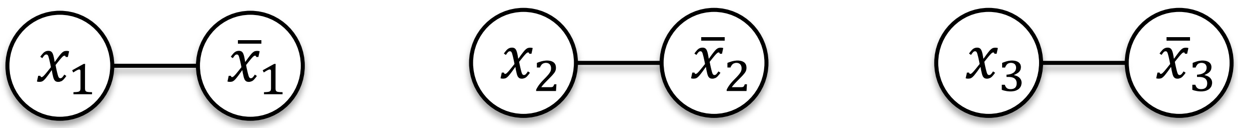
- Clause Gadget
  - ✓ For each clause C of ϕ and each literal x_i (or $\bar{x}_i$) in C, create one vertex and connect these vertices each other.



Examples two gadgets from a formula

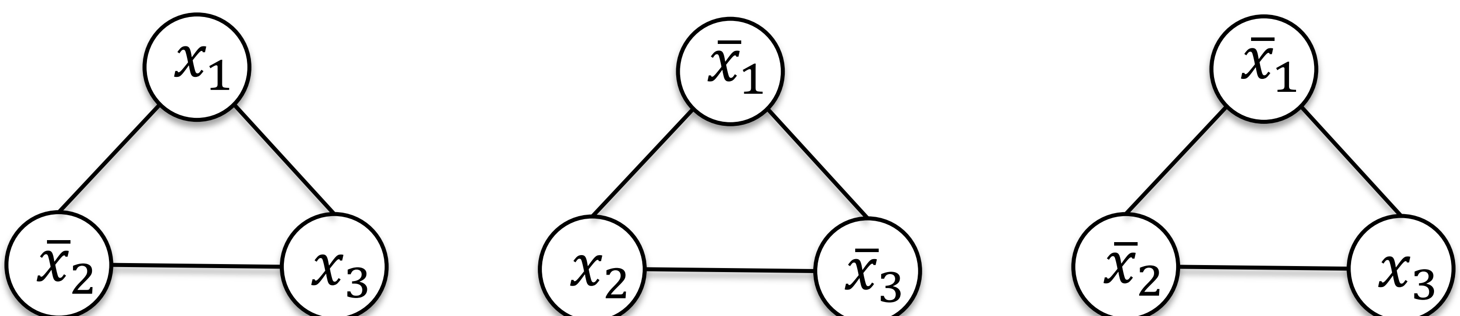
$$\phi = (x_1 \lor \bar{x}_2 \lor x_3)(\bar{x}_1 \lor x_2 \lor \bar{x}_3)(\bar{x}_1 \lor \bar{x}_2 \lor x_3)$$

Variable Gadets



Clause Gadets

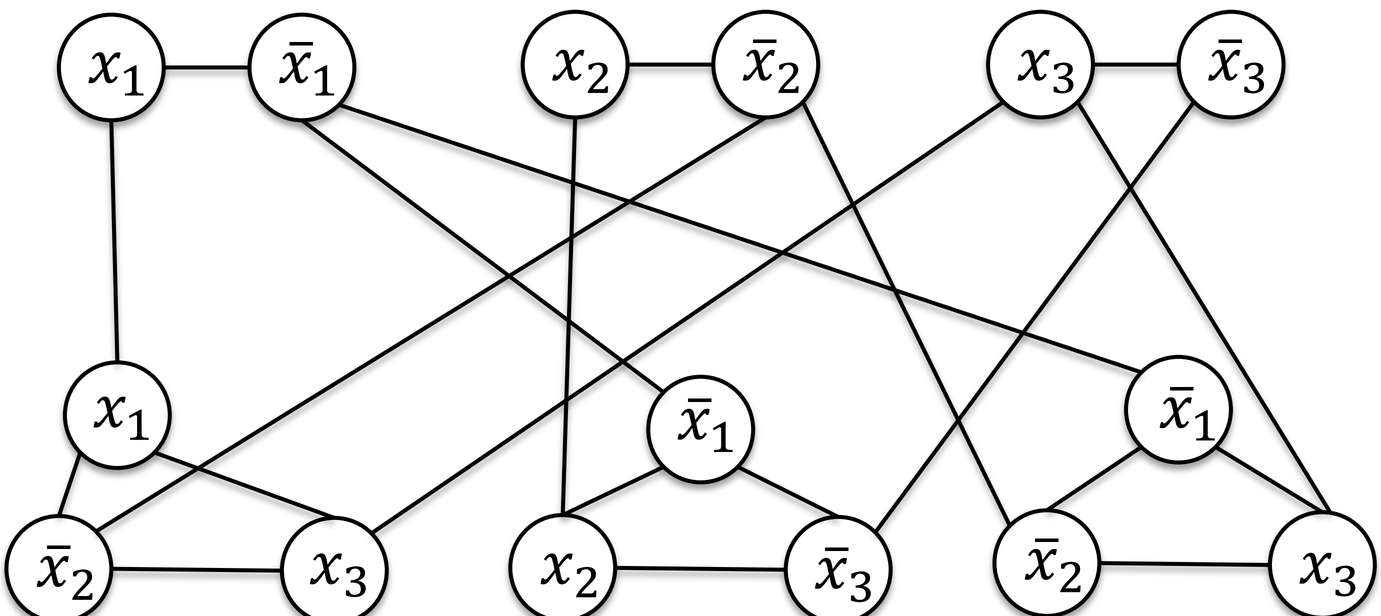
$(x_1 \lor \bar{x}_2 \lor x_3)$ $(\bar{x}_1 \lor x_2 \lor \bar{x}_3)$ $(\bar{x}_1 \lor \bar{x}_2 \lor x_3)$



A reduction from 3SAT to VC

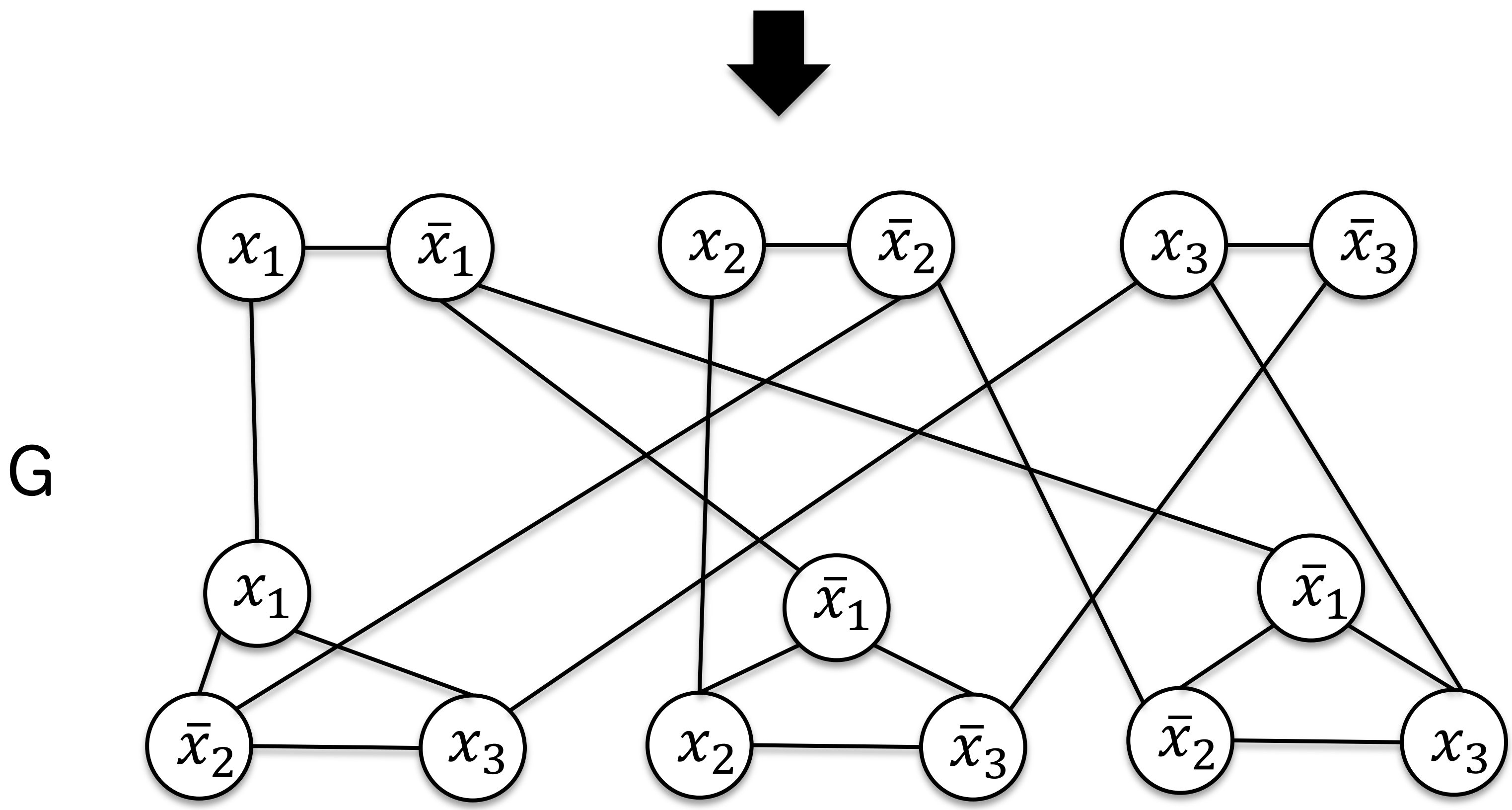
Next, we connect the variable gadgets to the clause gadgets.

Rule : the same label vertices are connected.



A reduction from 3SAT to VC

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



satisfying assignment to vertex cover

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

We consider a safisfying assignment α of ϕ such as $x_1 = 1, x_2 = 0, x_3 = 0$.

We obtain the desired vertex cover in the following manner.

- First, we take n vertices labeled by the literal that is set to 1.
  - ✓ In the above example, we take three vertices labeled with $x_1, \bar{x}_2, \bar{x}_3$

satisfying assignment to vertex cover

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

We consider a satisfying assignment α of ϕ such as $x_1 = 1, x_2 = 0, x_3 = 0$.

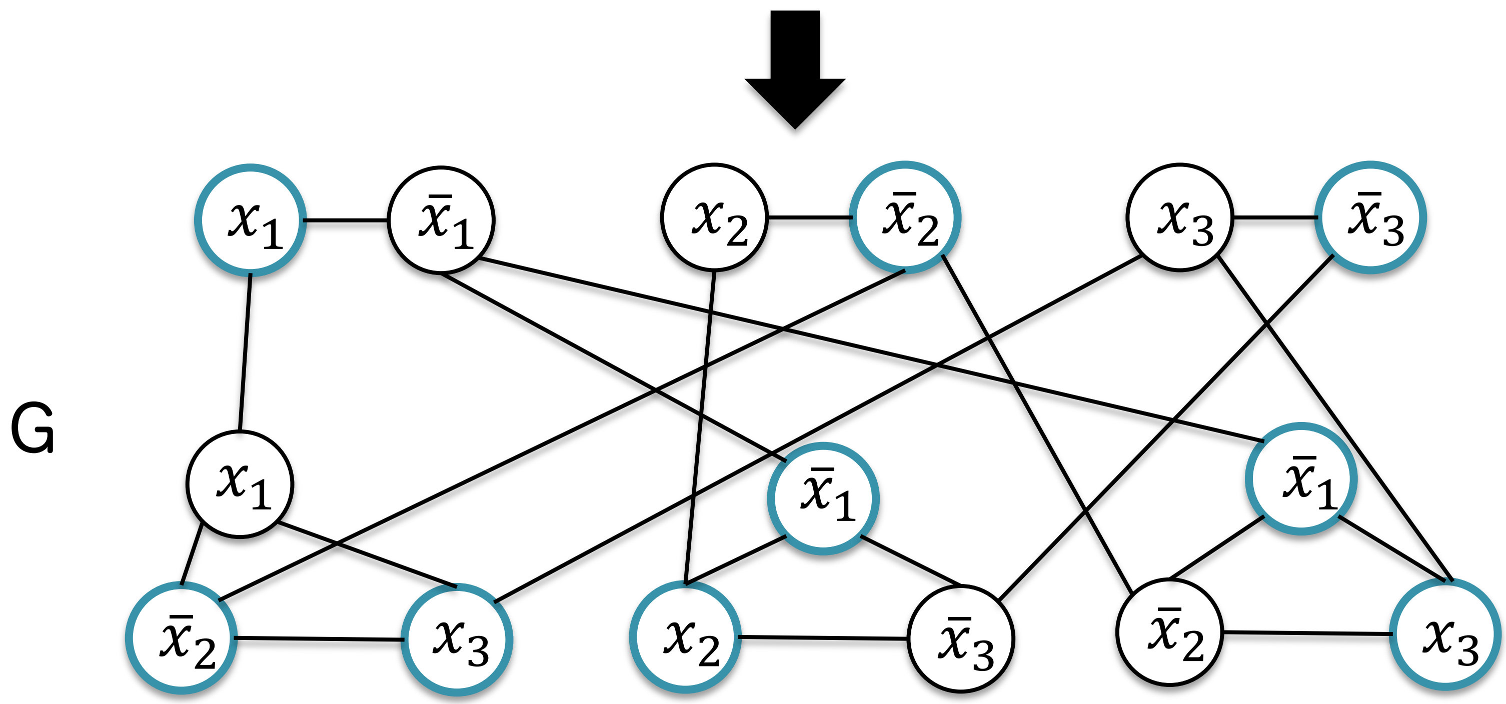
We obtain the desired vertex cover in the following manner.

- For each clause gadget, we take two vertices that is not connected to the vertex in variable gadgets taken in the previous slide. If there is no such vertex, we take arbitrary two vertices.

satisfying assignment to vertex cover

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

Satisfying assignment α : $x_1 = 1, x_2 = 0, x_3 = 0$



A set of blue vertices is a vertex cover, correctly.

satisfying assignment to vertex cover

Let ϕ be a 3CNF with n variables and m clauses.
Let G be the graph that is reducible from ϕ by the previous described reduction.

The number of vertices in G is 2n+3m and the number of edges in G is 4n+3m.
The size of VC is n+2m because for each variable gadget, one vertex is included in VC and for each clause gadget, two vertices is included in VC.

Why is this reduction correct?

Let ϕ be a 3CNF with n variables and m clauses.

Let G be the graph that is reducible from ϕ by the previous described reduction.

We need to prove the followings for correctness.

- The reduction runs in polynomial time.
- ($\Rightarrow$) If a given ϕ is satisfiable, then there exists a vertex cover of size n+2m in G.
- ($\Leftarrow$) If there exists a vertex cover of size n+2m in G, then ϕ is satisfiable.

Why is this reduction correct?

Let ϕ be a 3CNF with n variables and m clauses.

Let G be the graph that is reducible from ϕ by the previous described reduction.

We need to prove the followings for correctness.

- The reduction runs in polynomial time.
- ($\Rightarrow$) If a given ϕ is satisfiable, then there exists a vertex cover of size $n+2m$ in G.
- ($\Leftarrow$) If there exists a vertex cover of size $n+2m$ in G, then ϕ is satisfiable.

The reduction runs in polynomial time.

Creating each variable gadget takes O(1) time

Creating each clause gadget takes O(1) time

Thus, creating all gadgets takes O(n)+O(m) = O(n+m) time.

Connecting each variable gadget to clause gadgets takes O(nm) time because the number of vertrices in variable gadgets is 2n and that in edge gadgets is 3m.

The overall running time is O(nm+n+m) = O(nm).
O(nm) is polynomial of n+m, thus the reduction is in polynomial time.

Why is this reduction correct?

Let ϕ be a 3CNF with n variables and m clauses.

Let G be the graph that is reducible from ϕ by the previous described reduction.

We need to prove the followings for correctness.

- The reduction runs in polynomial time.
- (⇒) If a given ϕ is satisfiable, then there exists a vertex cover of size n+2m in G.
- (⇐) If there exists a vertex cover of size n+2m in G, then ϕ is satisfiable.

A sketch of proof for (⇒) part

There exists a satisfying assignment to α, then for each clause C, there exists at least one literal in C that is set to 1.

- For each variable gadget, one of two vertices is included in a vertex cover. Thus, all edges in variable gadgets are covered.
- For each clause gadget, two of three vertices are included in a vertex cover. Thus, all edges in clause gadgets are covered.

A sketch of proof for (⇒) part

It remains to show that any edge between variable gadgets and clause gadgets is covered.

- Any edge that is connected to the vertex in variable gadgets included in the vertex cover have already been covered.
- Other edges have also already covered by vertices in clause gadgets included in the vertex cover.

Why is this reduction correct?

Let ϕ be a 3CNF with n variables and m clauses.

Let G be the graph that is reducible from ϕ by the previous described reduction.

We need to prove the followings for correctness.

- The reduction runs in polynomial time.
- ($\Rightarrow$) If a given ϕ is satisfiable, then there exists a vertex cover of size n+2m in G.
- ($\Leftarrow$) If there exists a vertex cover of size n+2m in G, then ϕ is satisfiable.

A sketch of proof for (⇐) part

To obtain a vertex cover, we must take vertices while satisfying the following two conditions.

- Take at least one vertex from each variable gadget.
- Take at least two vertices from each clause gadget.

Now, we consider a vertex cover of size n+2m, thus we must take

- exact one vertex from each variable gadget
- exact two vertices from each clause gadget

A sketch of proof for ($\Leftarrow$) part

Our assumption assure that there exists a vertex cover of size n+2m, thus we can obtain the desired vertex cover C for taking vertices appropriately.

It is easy to see that we can obtain a satisfying assignment α to ϕ from the label of the vertices in vertex gadgets of C.

Exercise 2

Reduce the following instance ϕ of 3SAT to an instance G of VC by the reduction explained in the previous slides.

$$\phi = (x_1 \vee \bar{x}_3 \vee x_4)(\bar{x}_1 \vee x_2 \vee \bar{x}_3\ )(\bar{x}_2 \vee x_3 \vee \bar{x}_4)$$

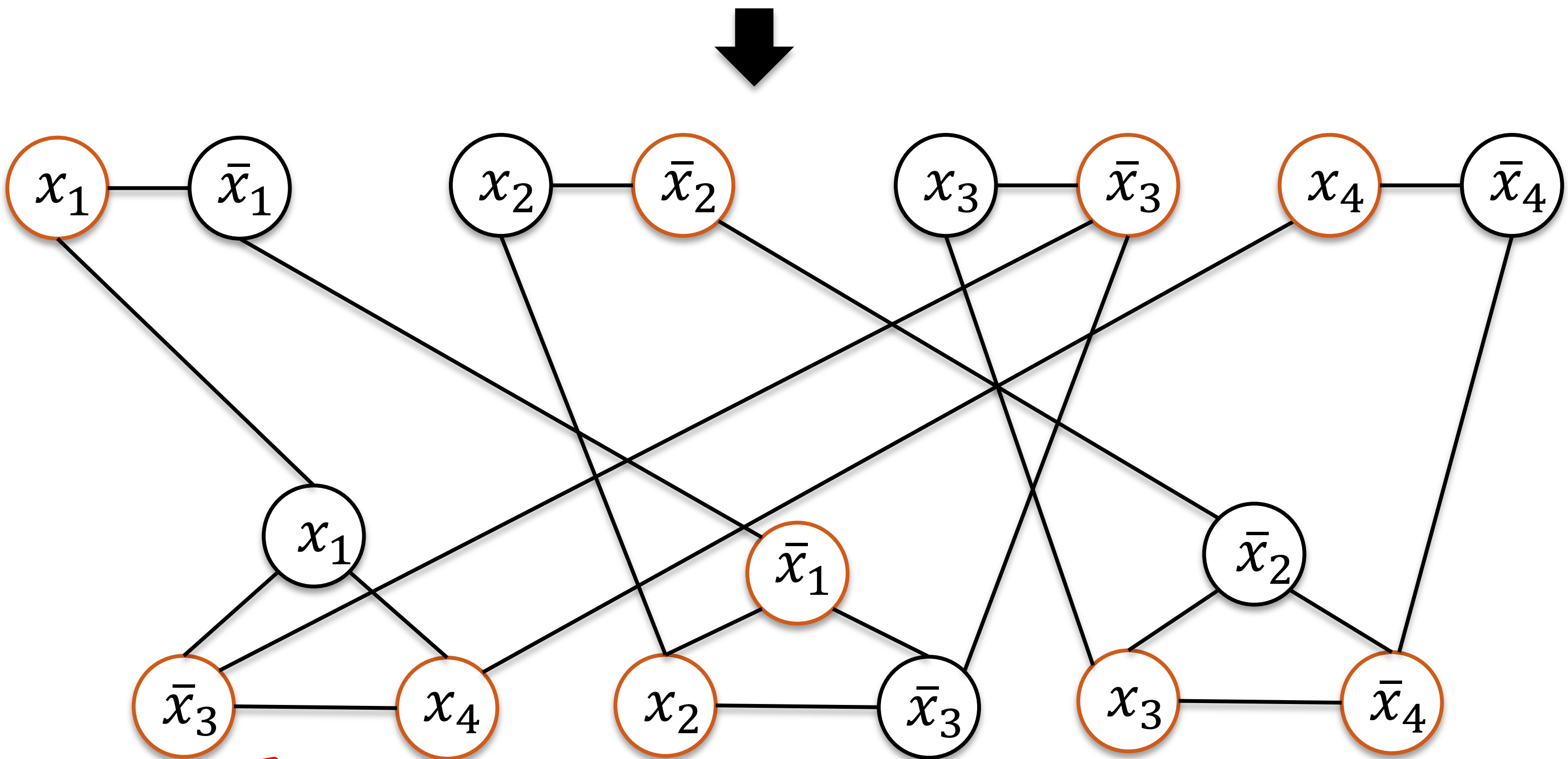
Let α be a satisfying assignment to ϕ such as

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

Show VC in G correspondings to α.

Exercise 2 (Answer)

$$\phi = (x_1 \vee \bar{x}_3 \vee x_4)(\bar{x}_1 \vee x_2 \vee \bar{x}_3)(\bar{x}_2 \vee x_3 \vee \bar{x}_4)$$



In this gadget,
take arbitrary two nodes.

Summary

We introduce the definition of NP-complete problem.

- Reductions and polynomial-time reductions
  - ✓ From the independent set problem to the vertex cover problem.
- NP-completeness via a polynomial-time reduction from CNF-SAT to 3SAT
- NP-completeness via a polynomial-time reduction from 3SAT to Vertex Cover