5800 Reductions

apr5/apr7 2022
shelat

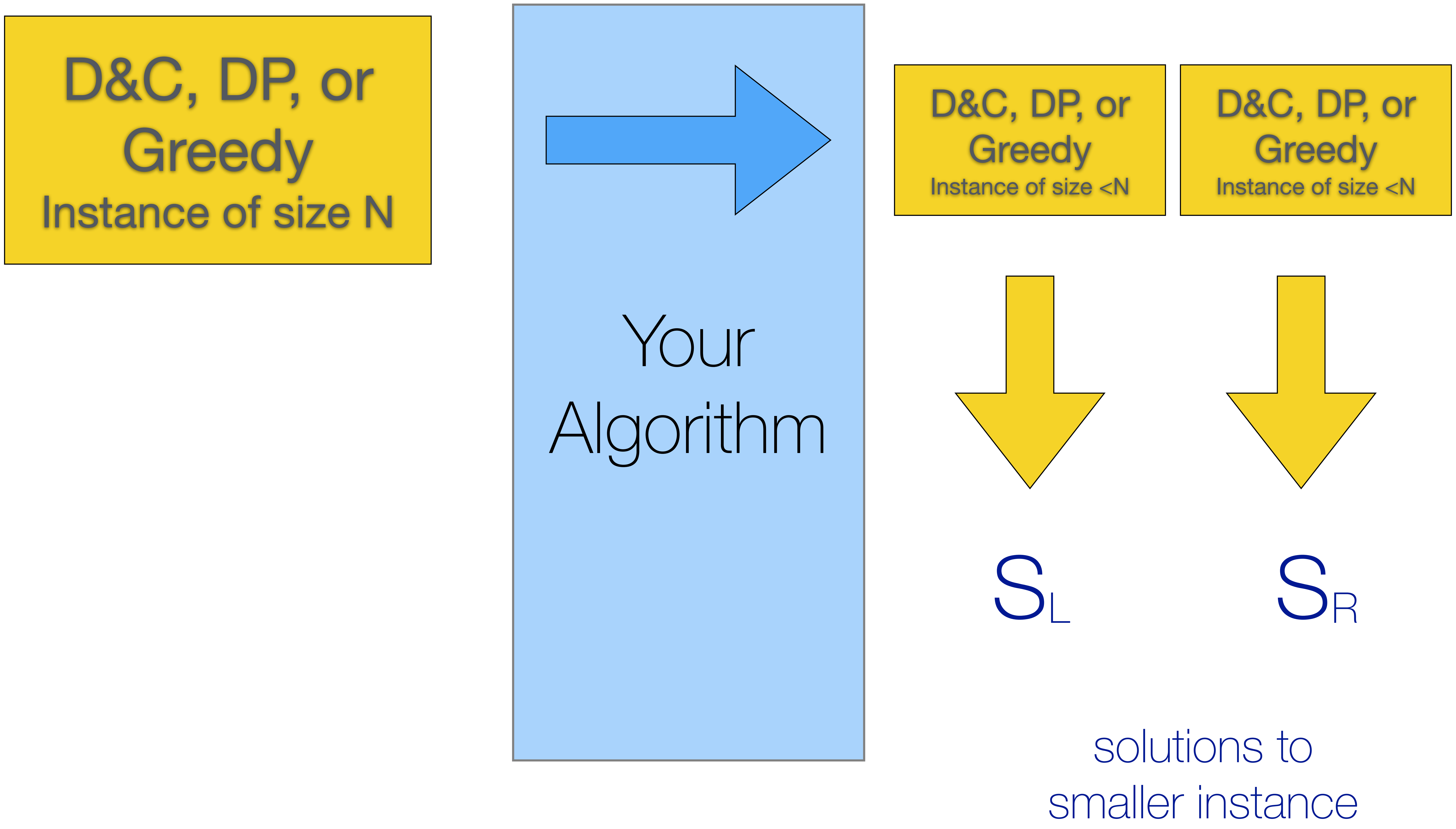
WE HAVE BEEN SOLVING PROBLEM A BY SOLVING SMALLER VERSIONS OF PROBLEM A

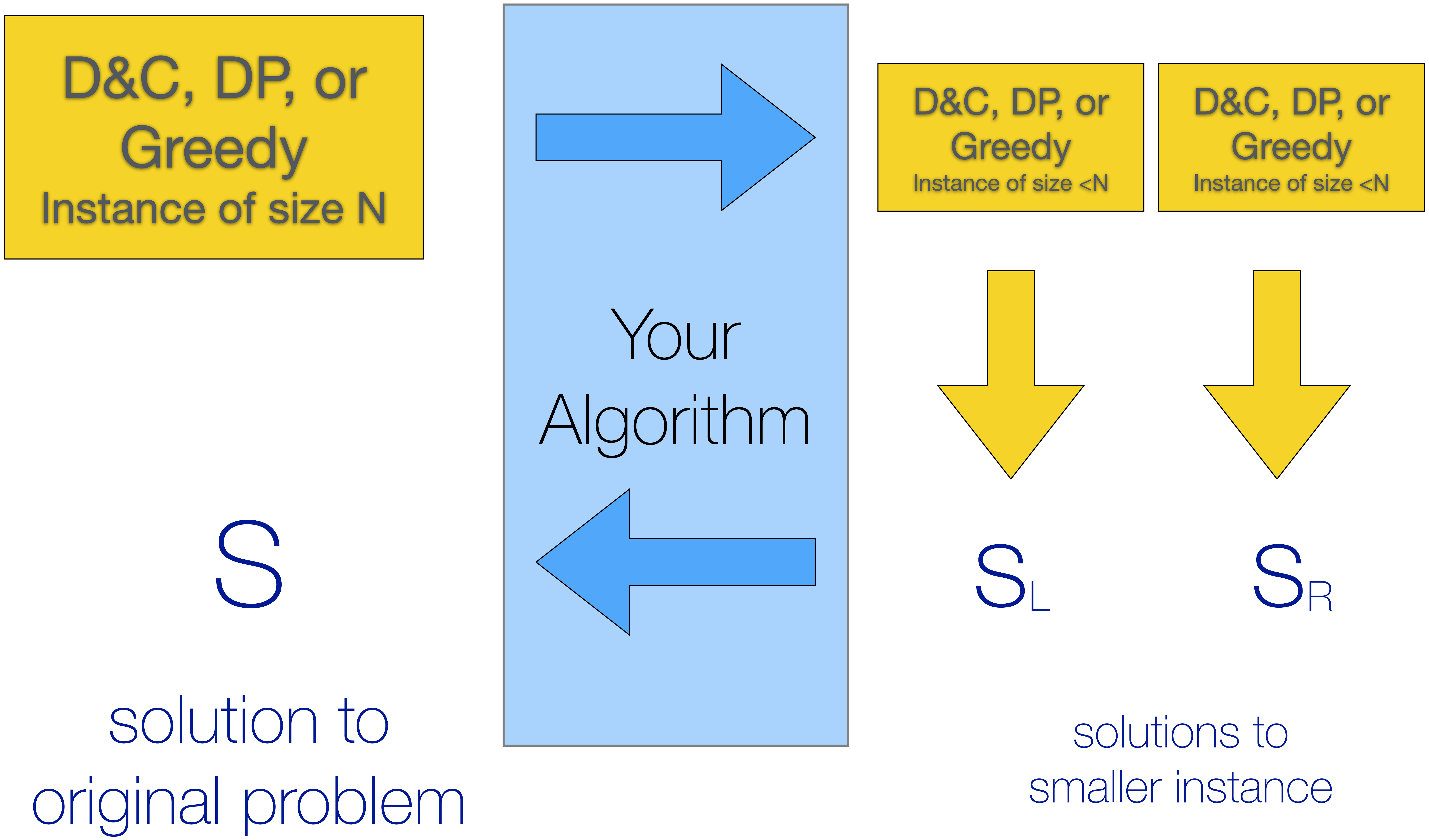
MORE GENERAL IDEA:

SOLVE PROBLEM A BY
SOLVING PROBLEM B

D&C, DP, or Greedy

Instance of size N


D&C, DP, or Greedy
Instance of size N
Your Algorithm
D&C, DP, or Greedy
Instance of size <N
D&C, DP, or Greedy
Instance of size <N
S_L
S_R
solutions to smaller instance


D&C, DP, or Greedy
Instance of size N
Your Algorithm
D&C, DP, or Greedy
Instance of size <N
D&C, DP, or Greedy
Instance of size <N
S
solution to original problem
S_L
S_R
solutions to smaller instance

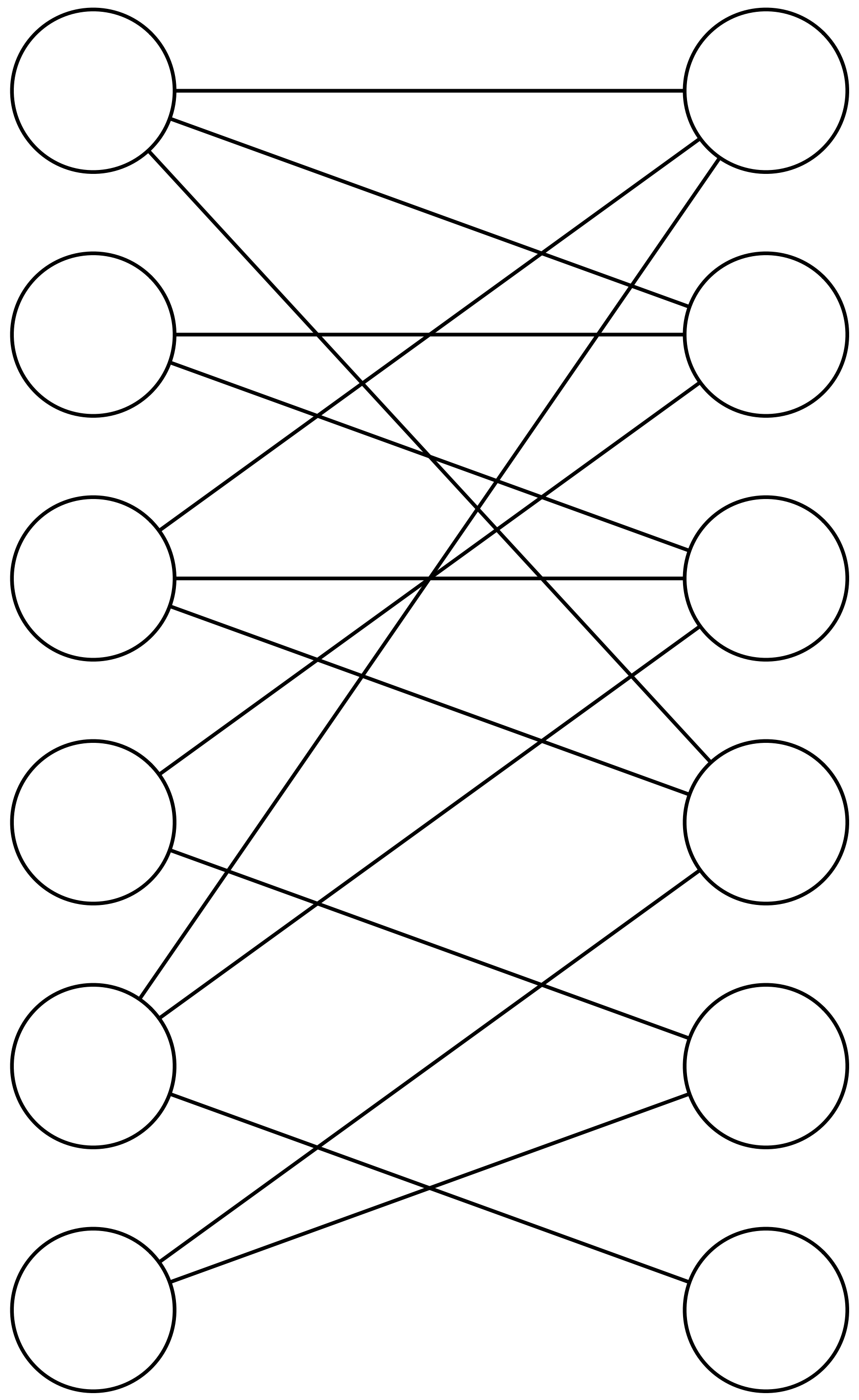
Bipartite Matching Algorithm

BP(L,R,E)

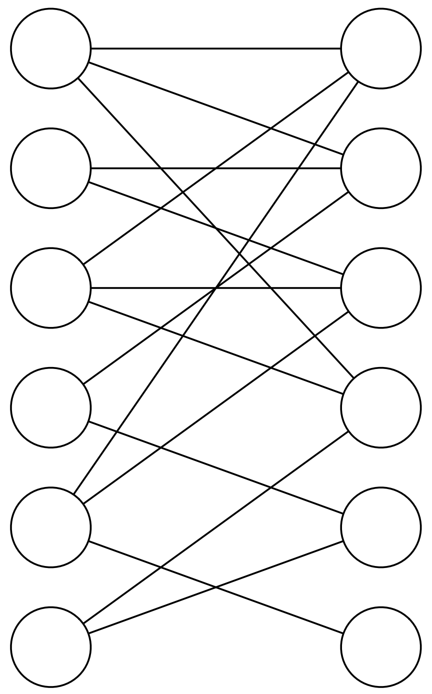
1. MAKE NEW G'
FROM INPUT G.

2. RUN FF ON G'

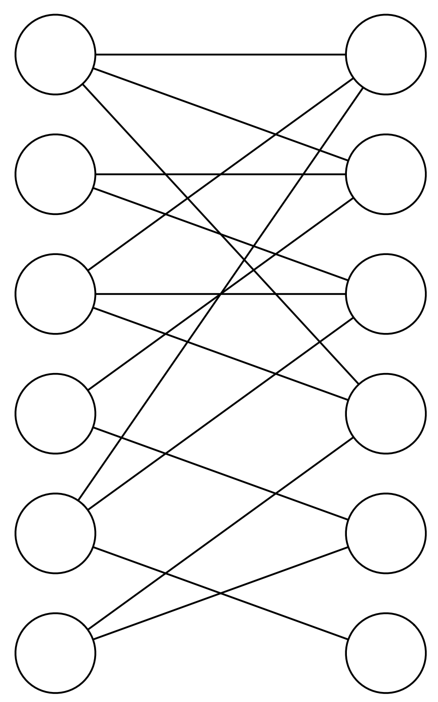
3. OUTPUT ALL MIDDLE EDGES
WITH FLOW F(E)=1.



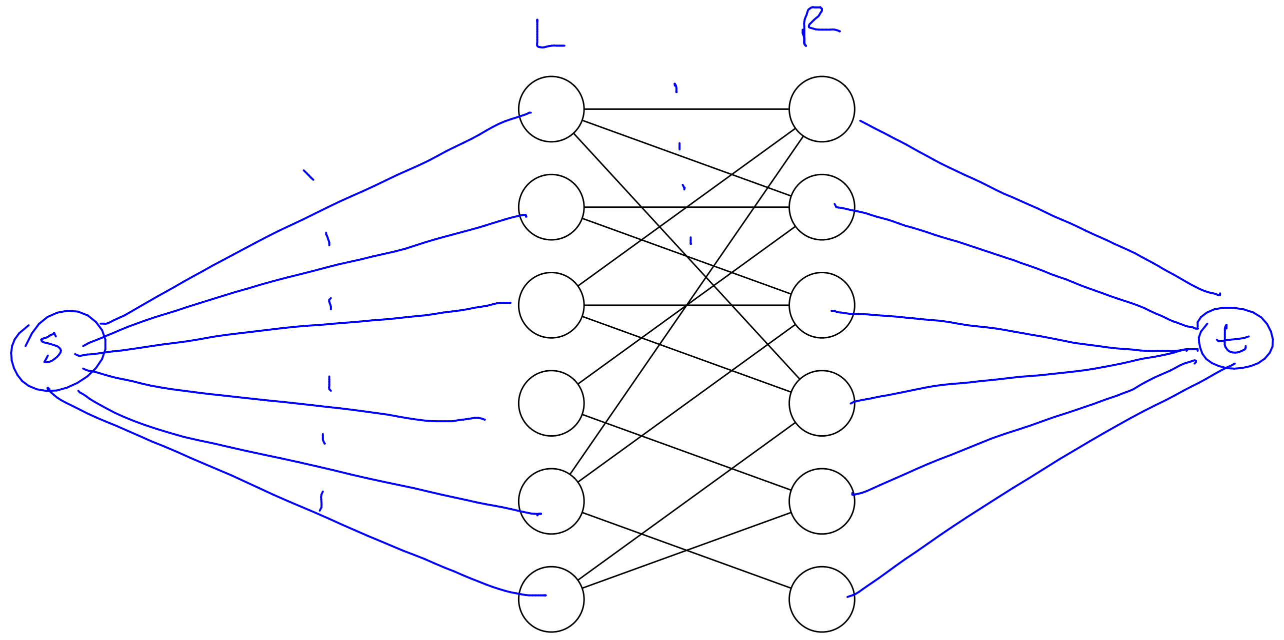
Bipartite Matching Instance



Bipartite Matching Instance



Max flow Instance



IF G HAS A MATCHING OF SIZE K, THEN G' has a MAXFLOW of K.

Proof: (outline) Let M^* be the matching of size K for G,

Construct flow f to be

$f(e) = 1$ if $e \in M^*$, and if $e = (x, y)$ then $f(s, x) = 1$, $f(y, t) = 1$

$\Rightarrow$ flow f satisfying ① capacity constraints

② flow constraint

INFLOW(x) = OUTFLOW(x)

G'

' HAS A FLOW OF K, THEN G HAS K-MATCHING.

(1) Our algorithm uses FF, so flow for G' is integral.

Now define $M: \{ e \mid f(e)=1 \text{ and } e=(x,y) \text{ s.t. } x \in L,\ y \in R \}$

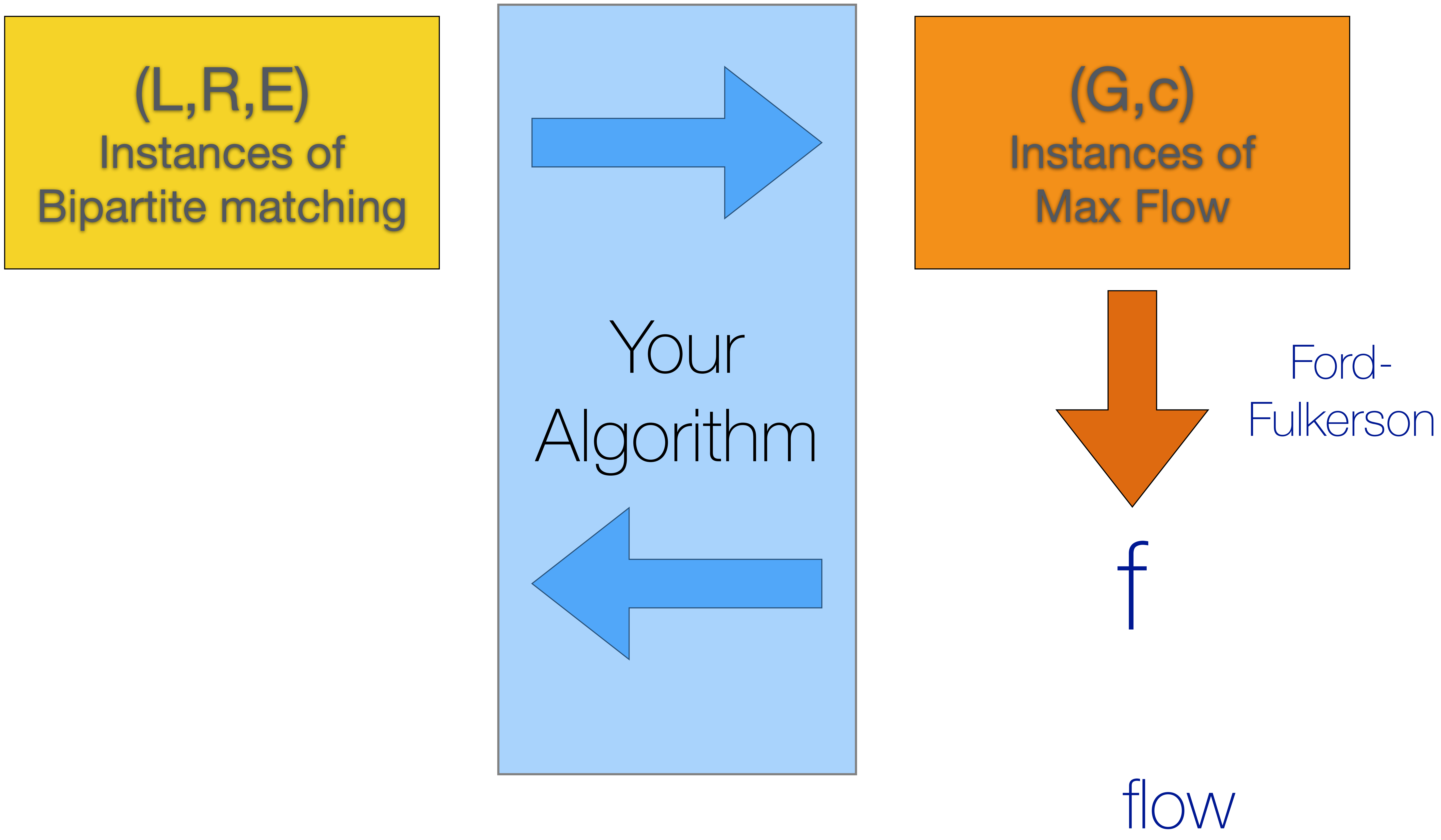
Prove that M is a matching.

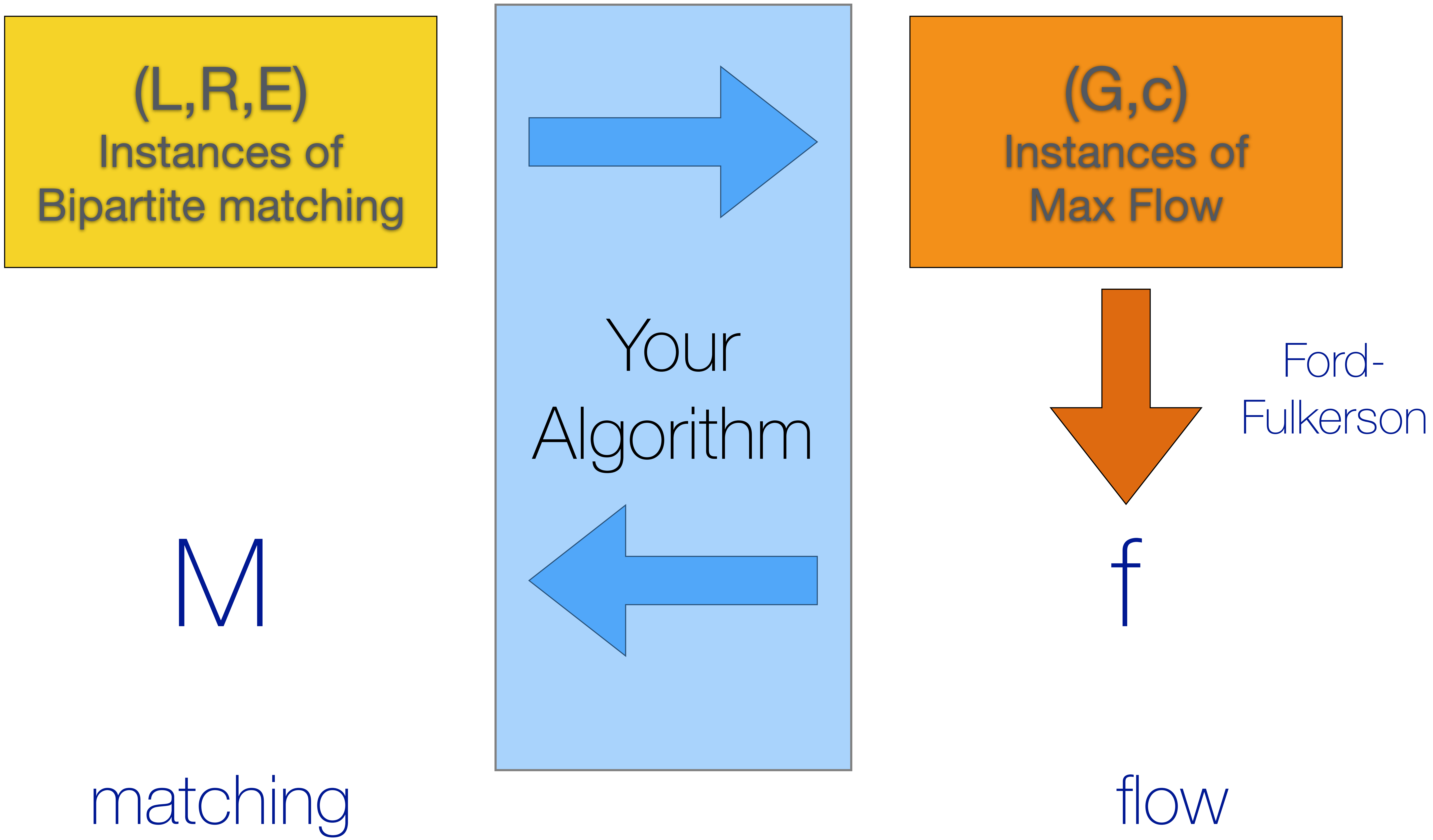
- All flows are integral & capacity $\underline{c(e)=1}$. so $f(e) = \underline{0}$ or $\underline{1}$ for $e \in E$.

Thus for all $\underline{v \in L}$ ($v \in R$), v is incident to at most 1 edge in M.

By the flow constraint. By MIN. cuts, $|M| = K$

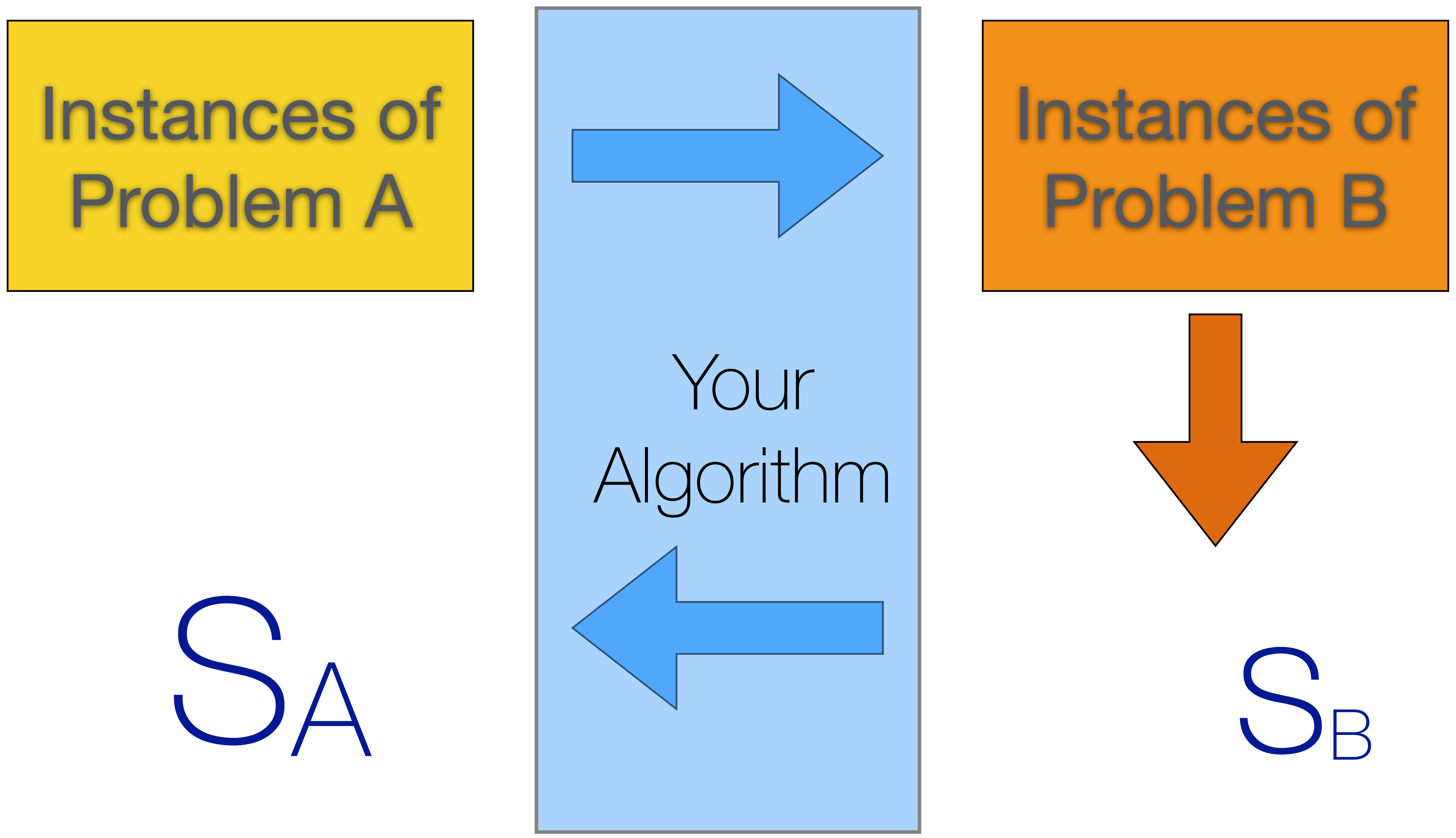
(L,R,E)
Instances of
Bipartite matching


(L,R,E)
Instances of
Bipartite matching
Your
Algorithm
(G,c)
Instances of
Max Flow
Ford-
Fulkerson
f
flow


(L,R,E)
Instances of
Bipartite matching
Your
Algorithm
(G,c)
Instances of
Max Flow
Ford-
Fulkerson
M
f
matching
flow

Reduction

$$\mathrm{PROBLEM}_a \leq_{f(n)} \mathrm{PROBLEM}_b$$



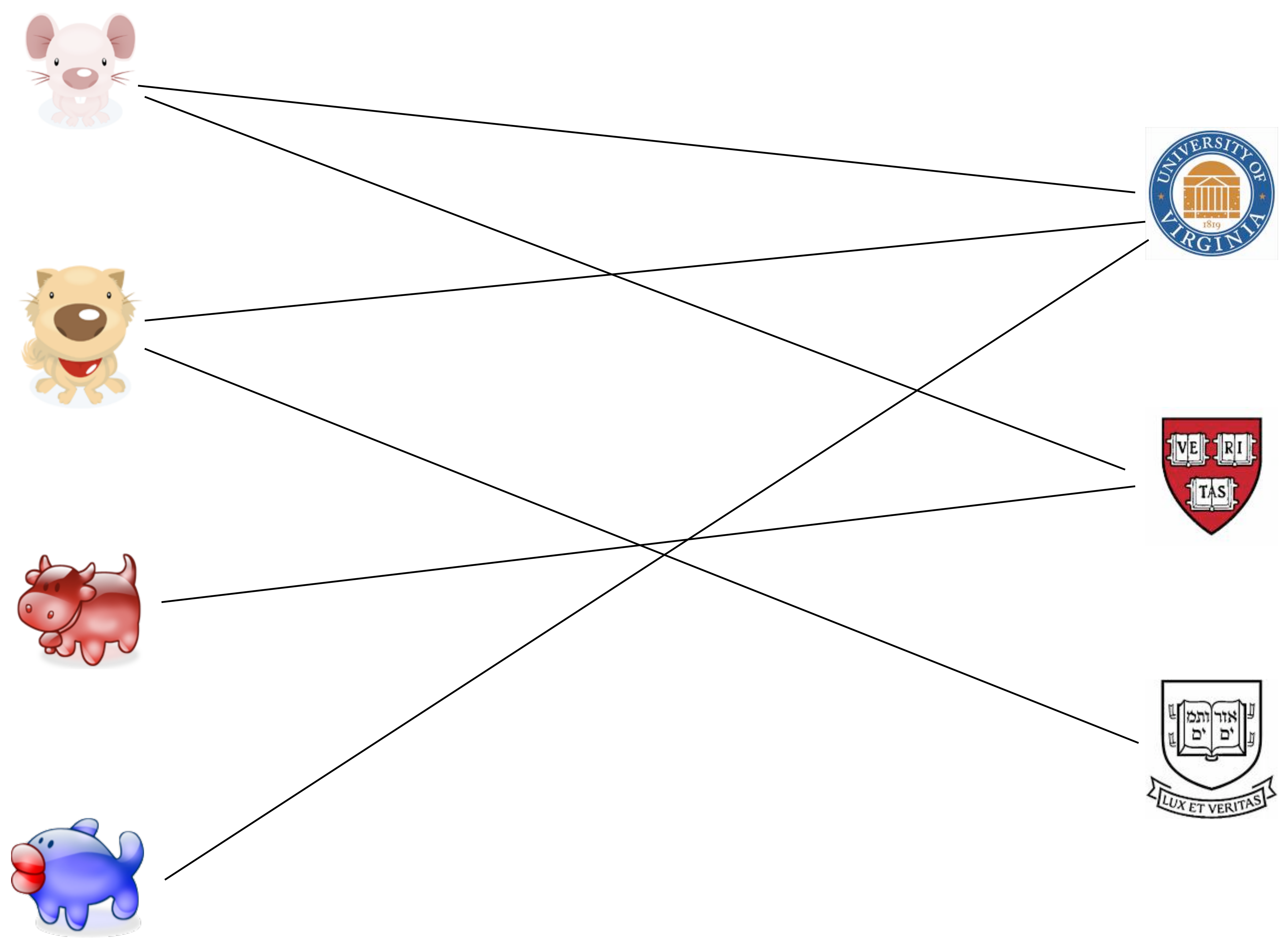
Runs in time f(n)

$$\text{PROBLEM}_a \leq_{f(n)} \text{PROBLEM}_b$$

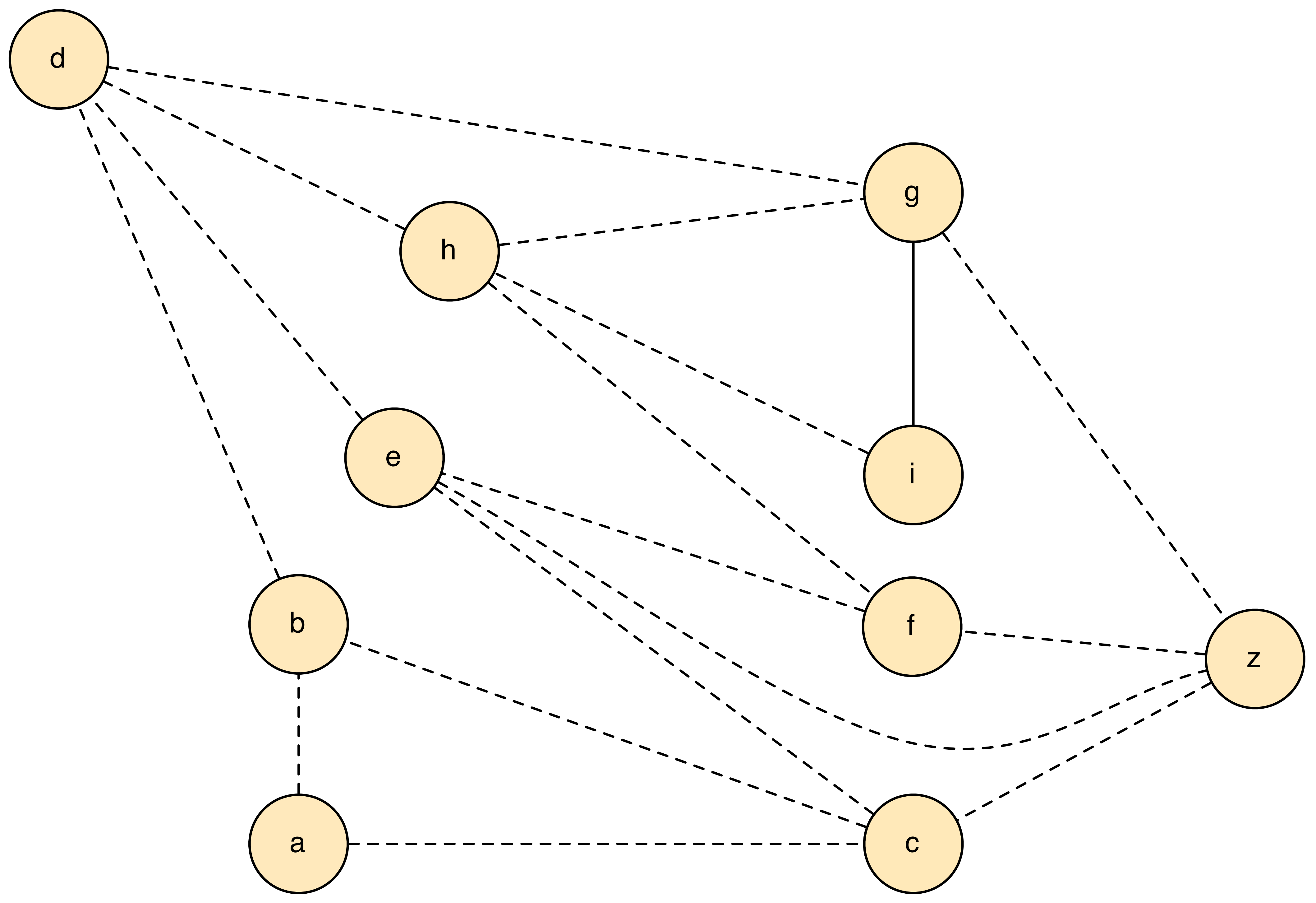
$$\exists c, d$$

$$T(\textsc{problem}_a(n)) \leq f(n) + cT(\textsc{problem}_b(dn))$$

Maximum bipartite



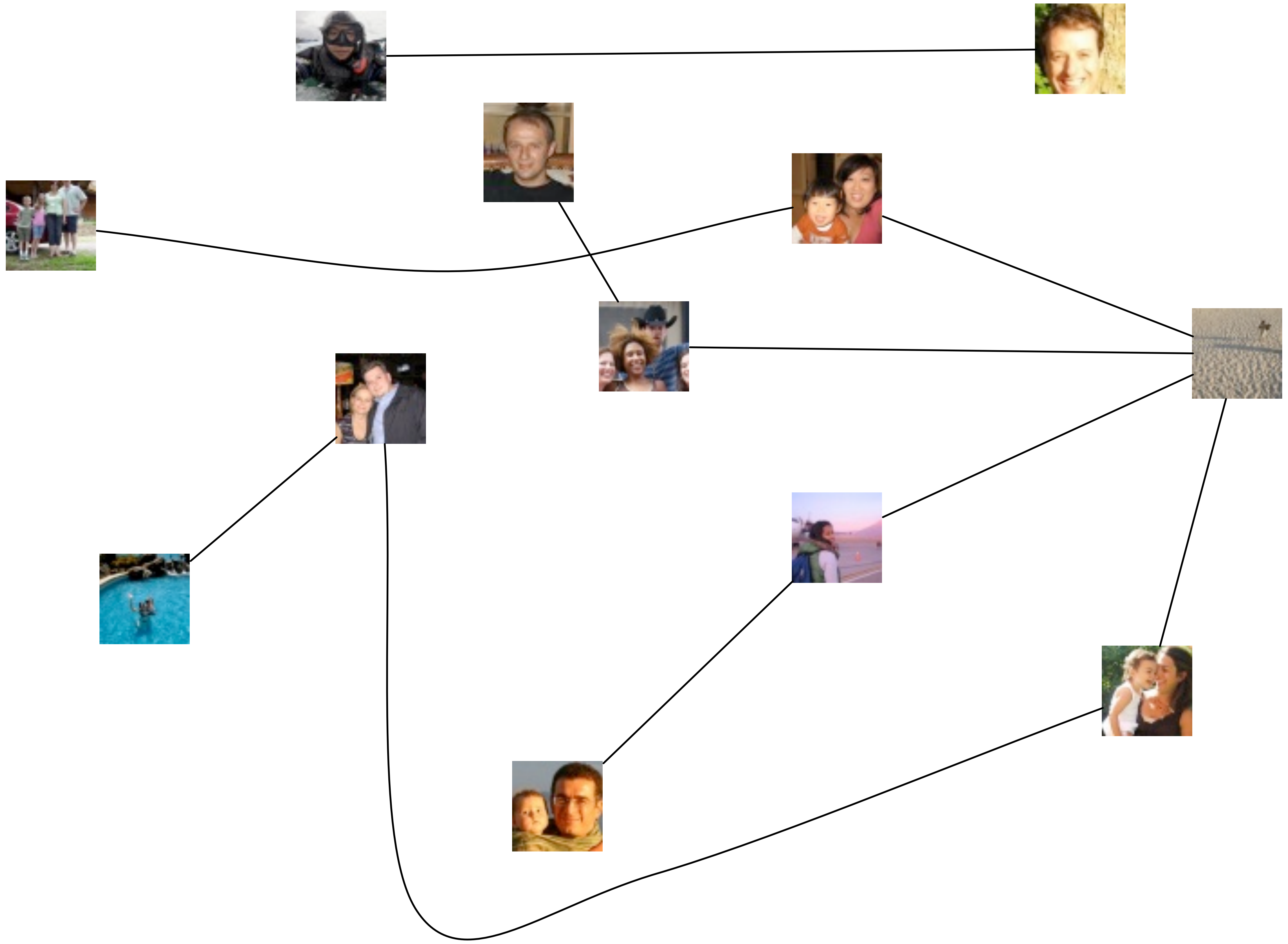
edge-disjoint paths



$$\text{MAXBIPARTITE} <_{E+V} \text{MAXFLOW}$$

$$\text{MAXEDGEDISJ} <_{E+V} \text{MAXFLOW}$$

party problem



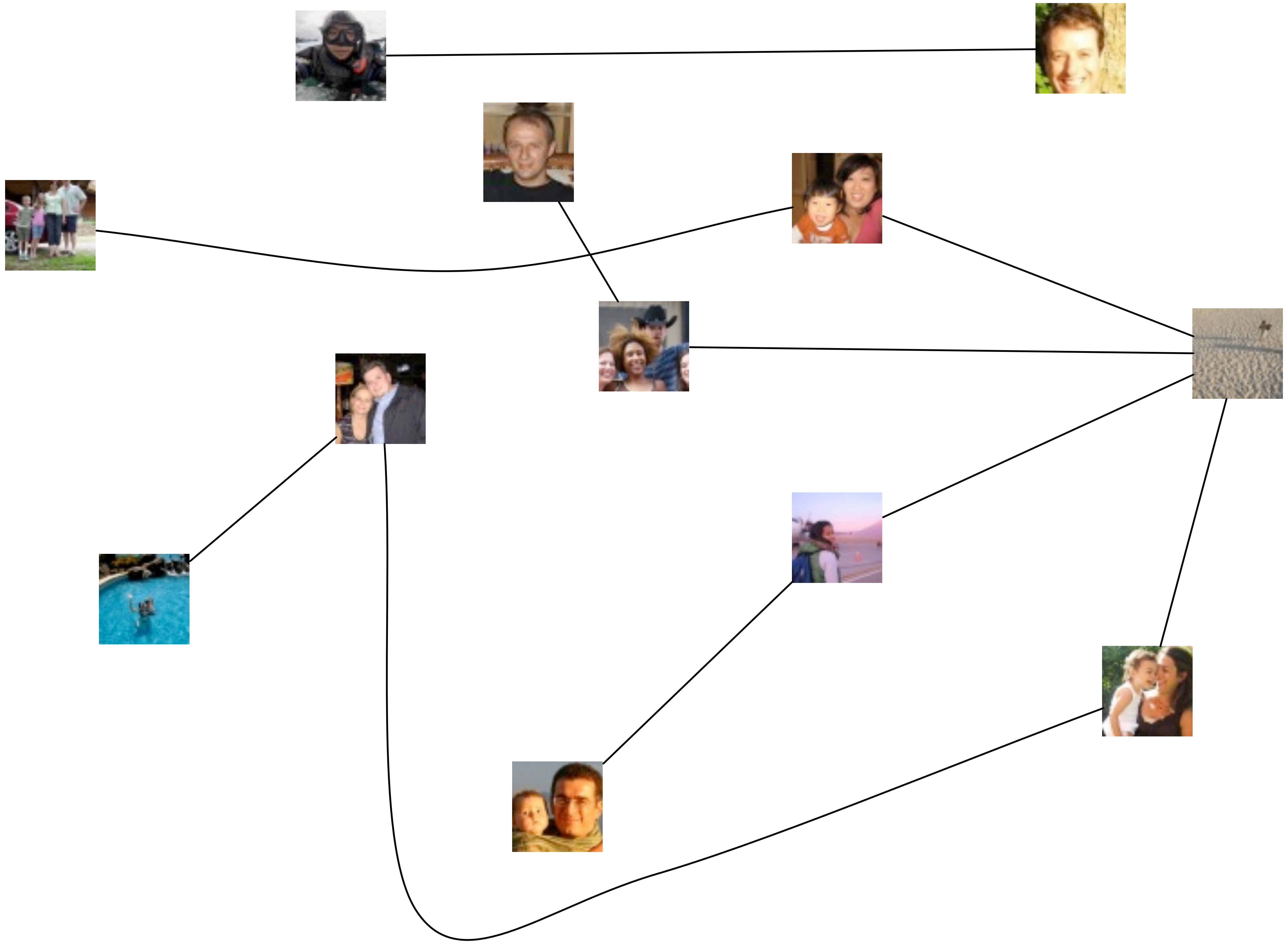
Graph of friends who do not get along with one another

Def: independent set

Def: independent set

Def: For a graph G, a set $S \subseteq V$ is an independent set if no two nodes in S are joined by an edge.

example



goal:

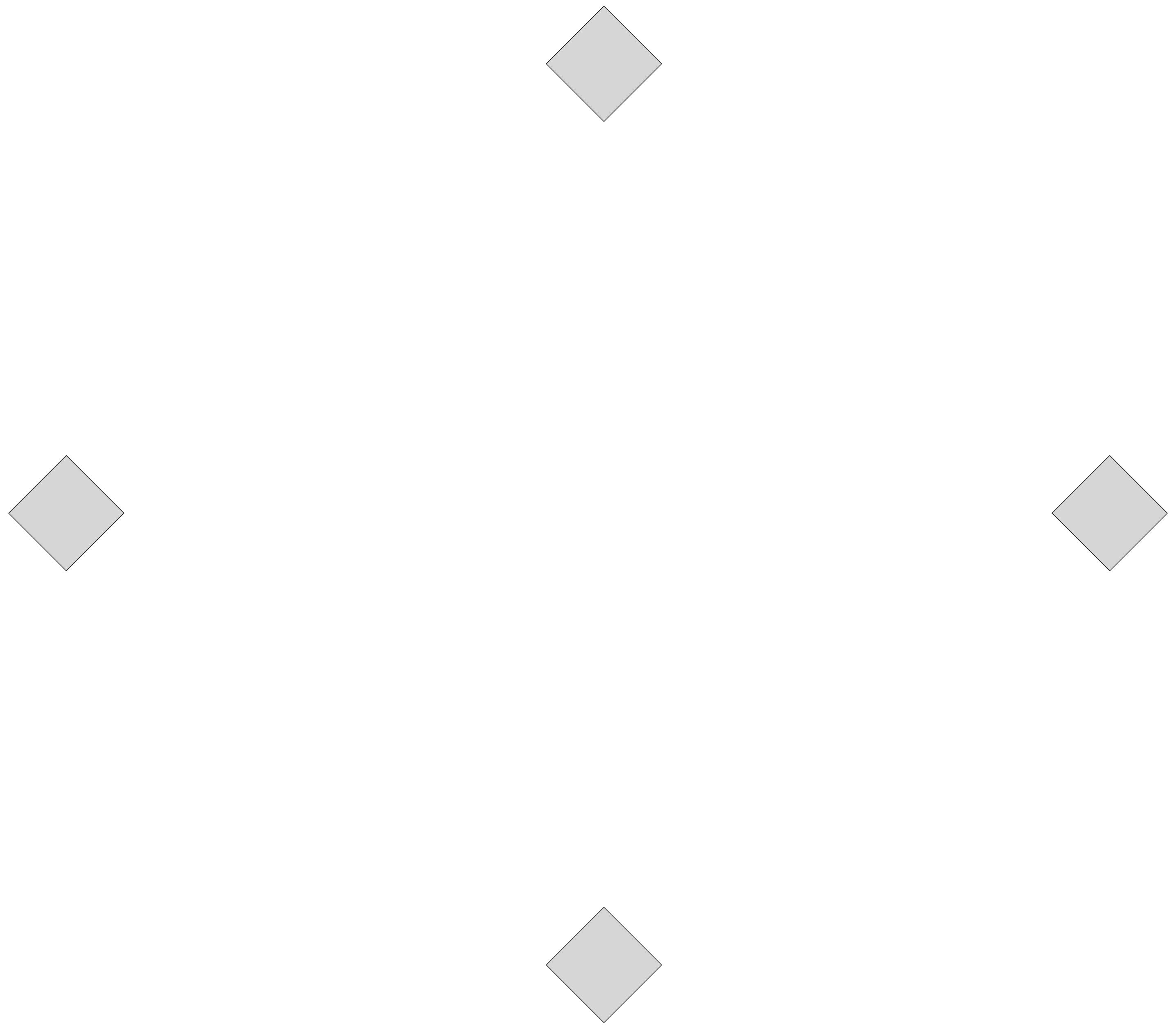
Given a graph $G=(V,E)$,

goal:

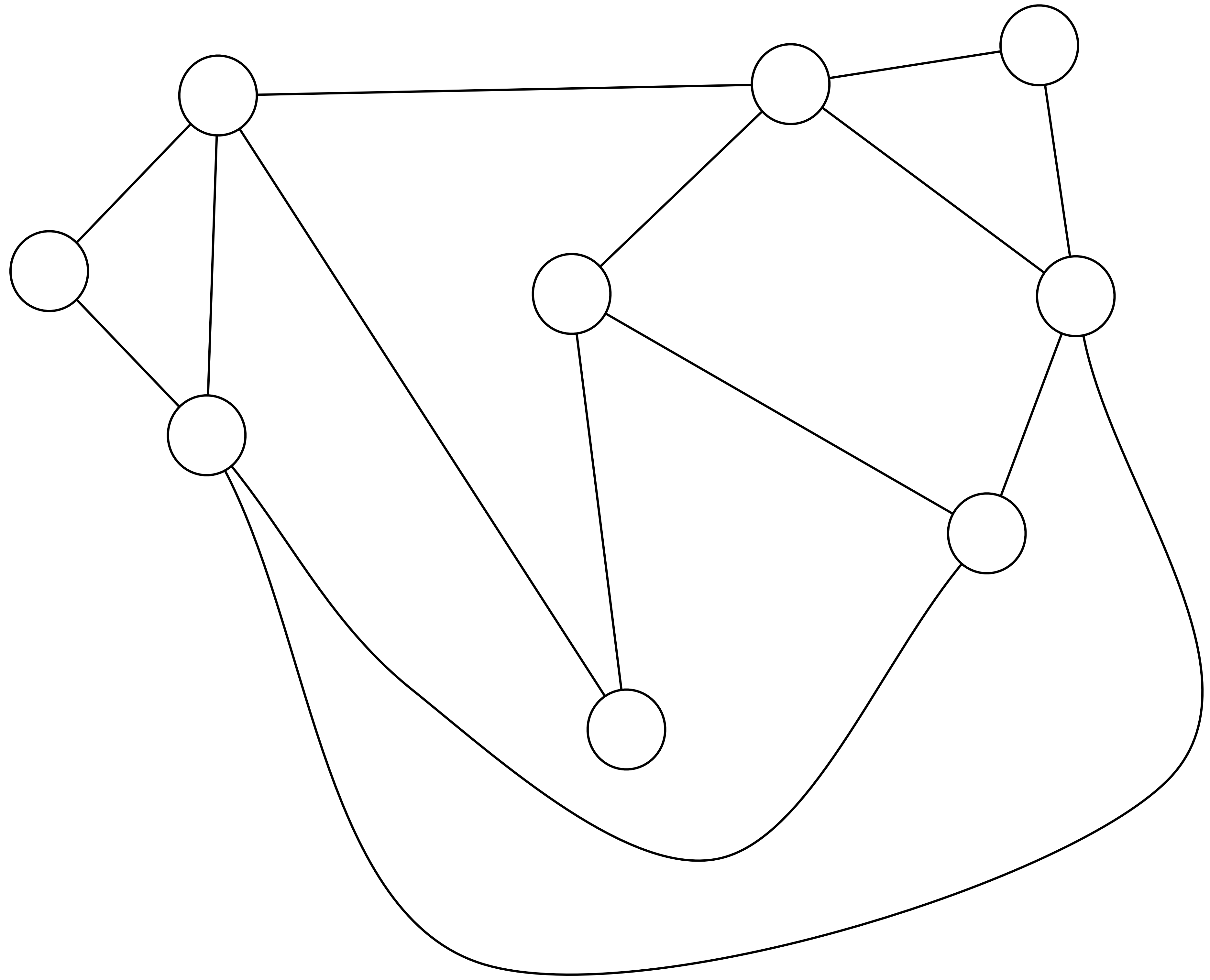
Given a graph G=(V,E), find the largest or max independent set.

This represents the largest group of people who are conflict free.

baseball



baseball



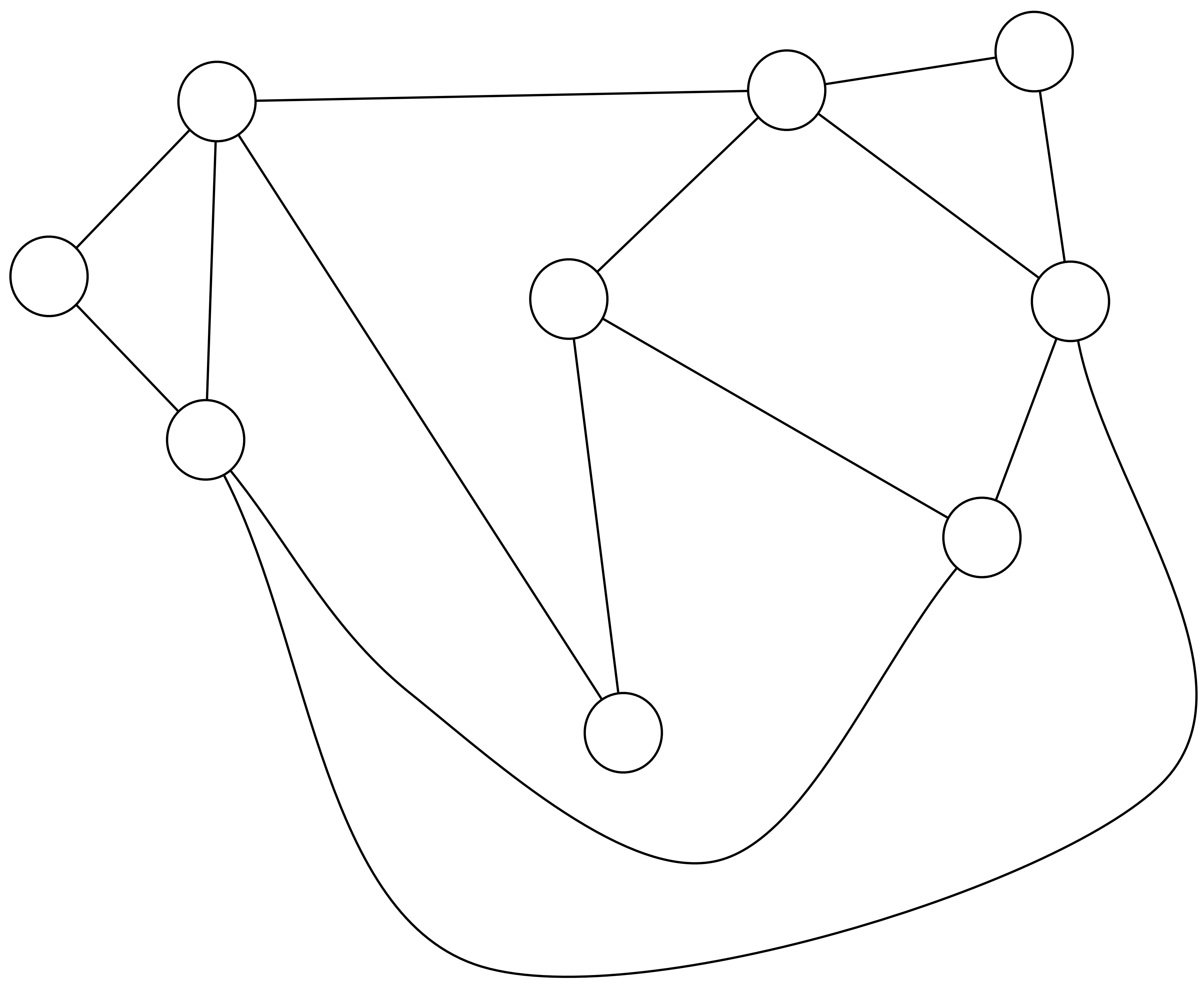
Imagine a scalable, abstract version of baseball or “n” players.

A vertex cover of a graph G=(V,E) is a

A vertex cover of a graph G=(V,E) is a

Set of nodes S such that for each edge $e = (x, y) \in E$, either $x \in S$ or $y \in S$.

example



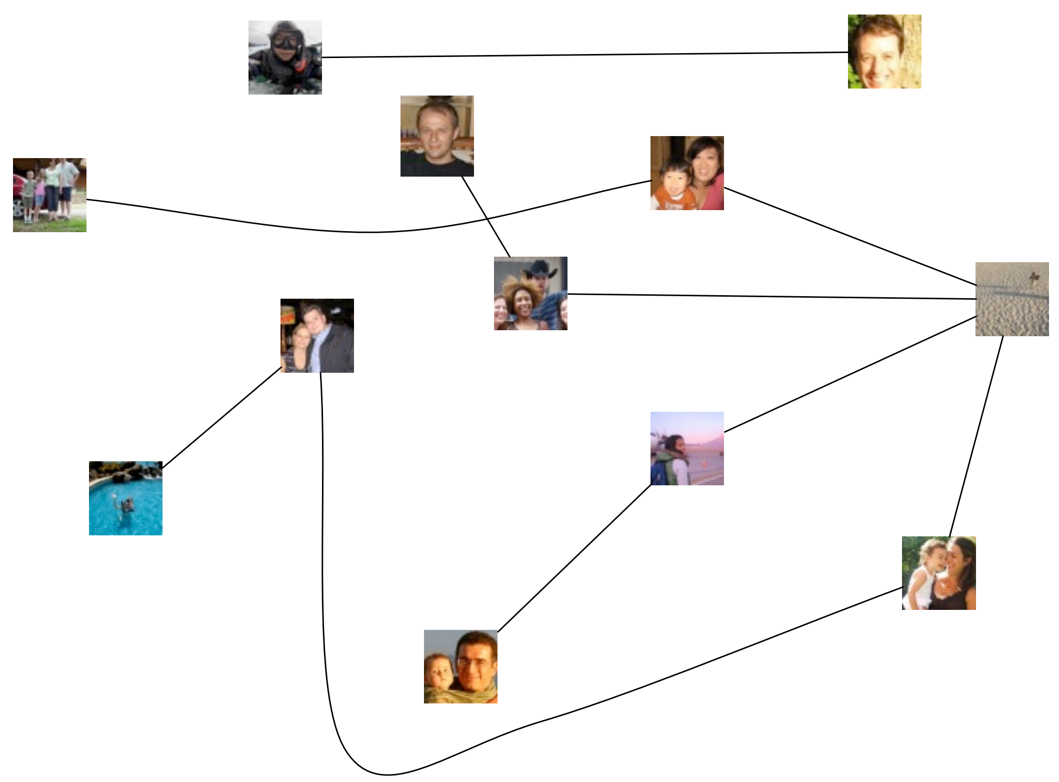
goal:

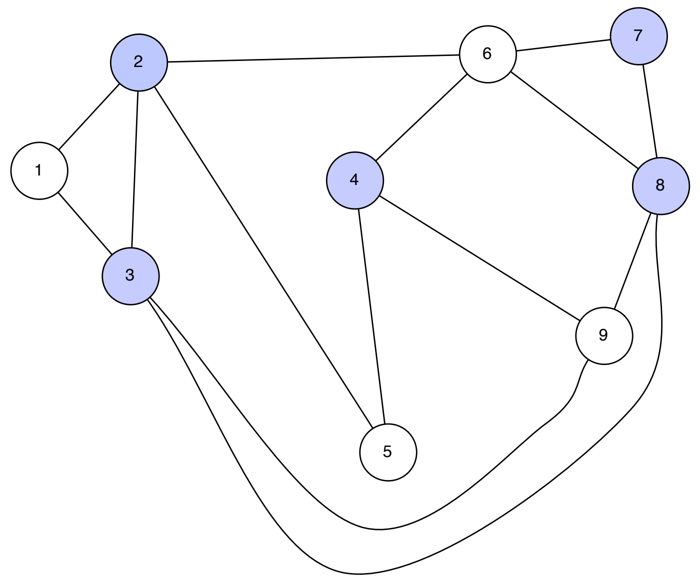
given a graph G,

goal:

given a graph G,

Find the minimum sized vertex cover for G.





$$\text{MAXINDSET} \leq_{O(V)} \text{MINVERTEXCOVER}$$

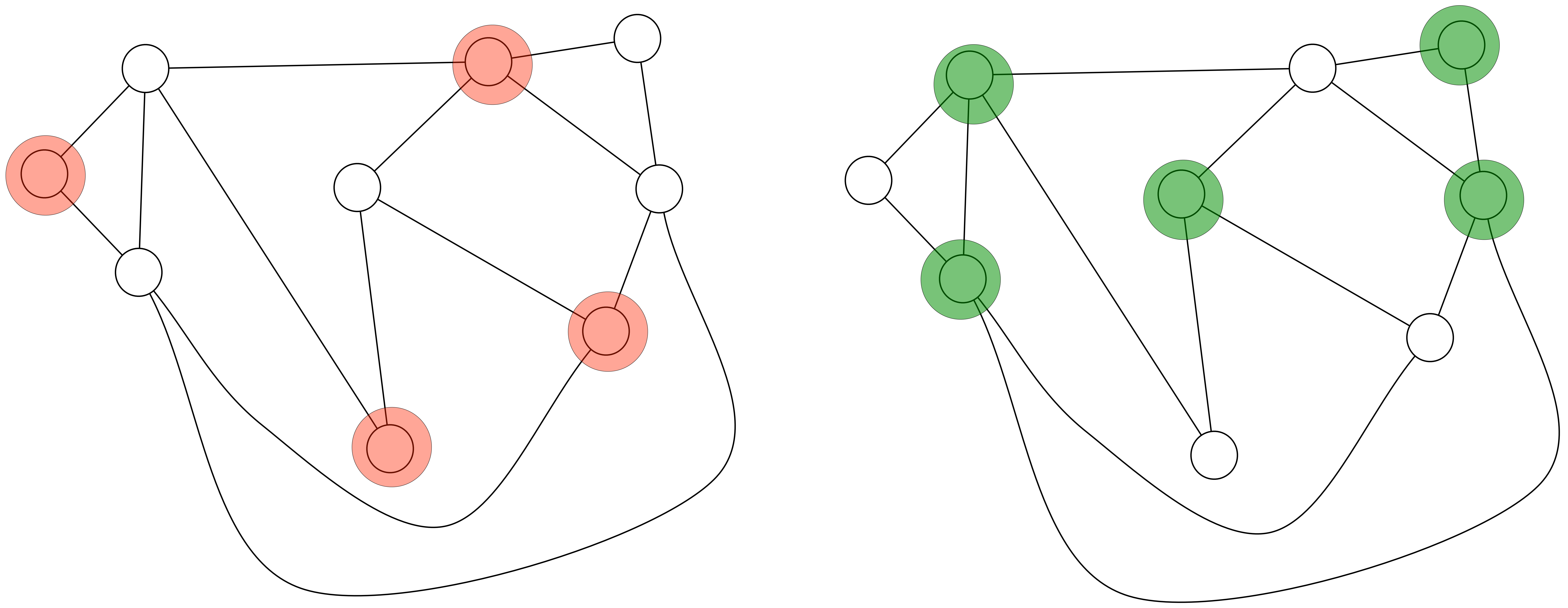
A solution to VC can be used to solve INDSET.

What is required to show this reduction?

Thm:

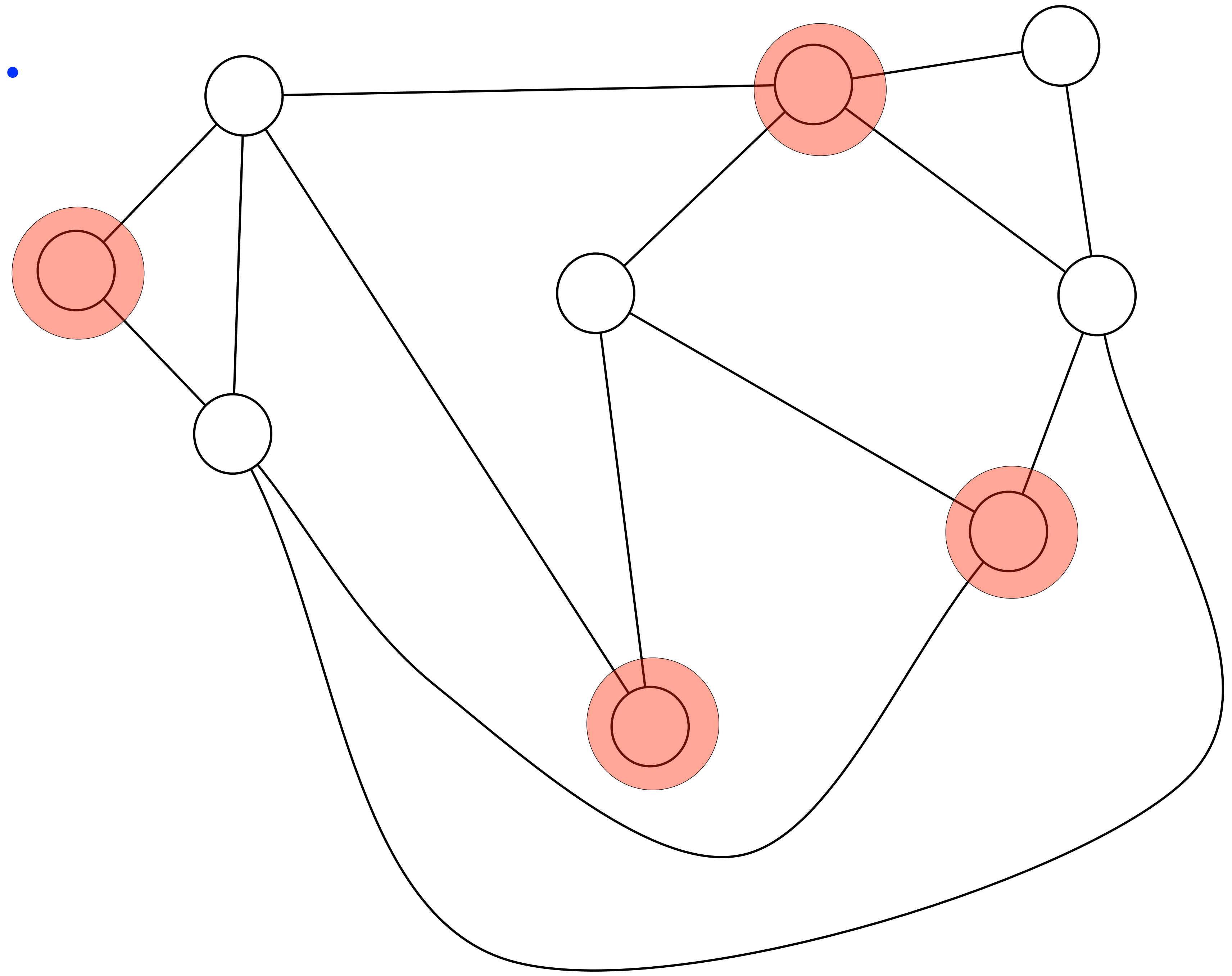
Thm: For a graph G=(V,E), S is a vertex cover of G if and only if (V-S) is an independent set of G.

Thm: set S is an independent set of G iff V-S is a vertex cover.



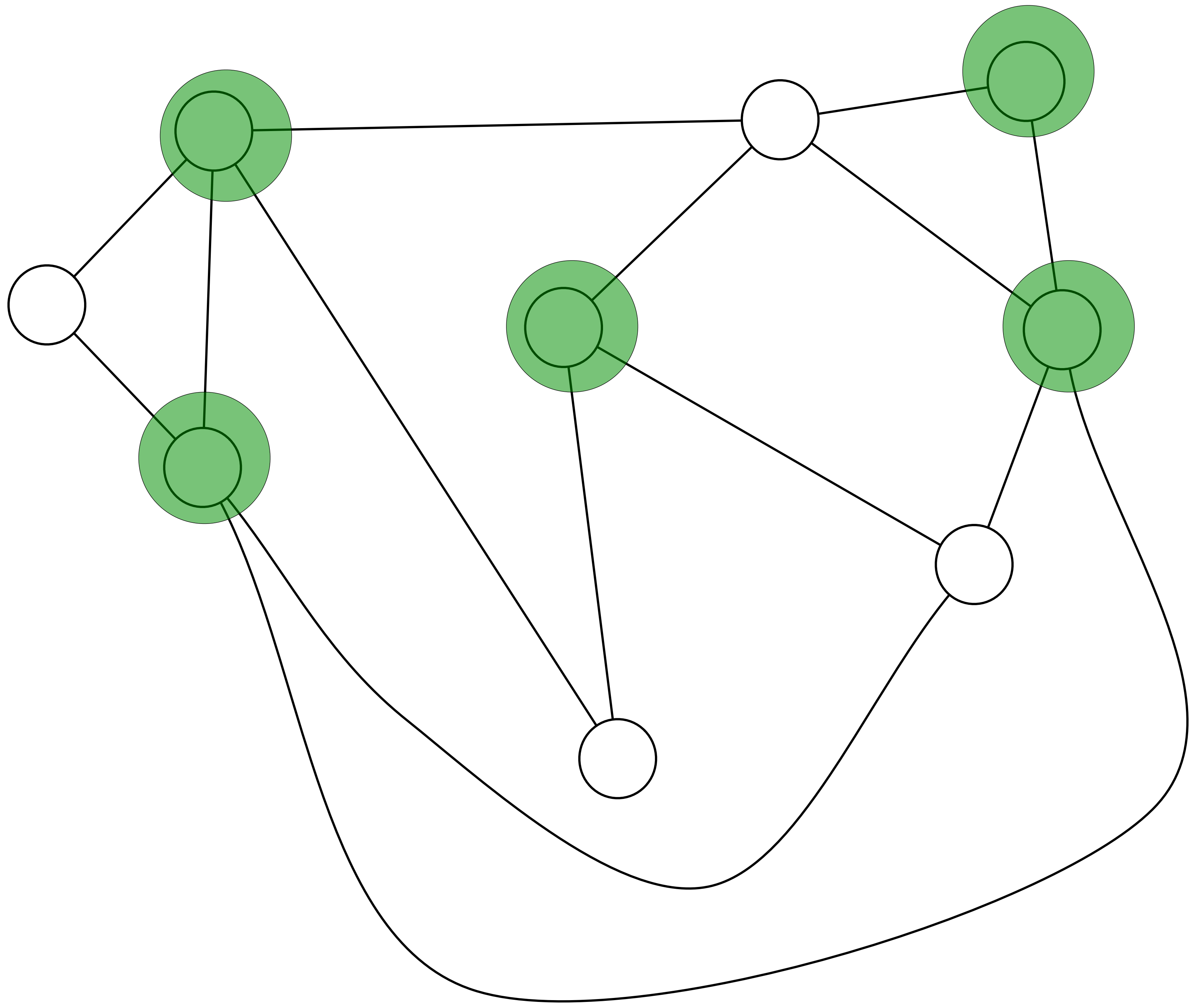
Thm: set S is an independent set of G iff V-S is a vertex cover.

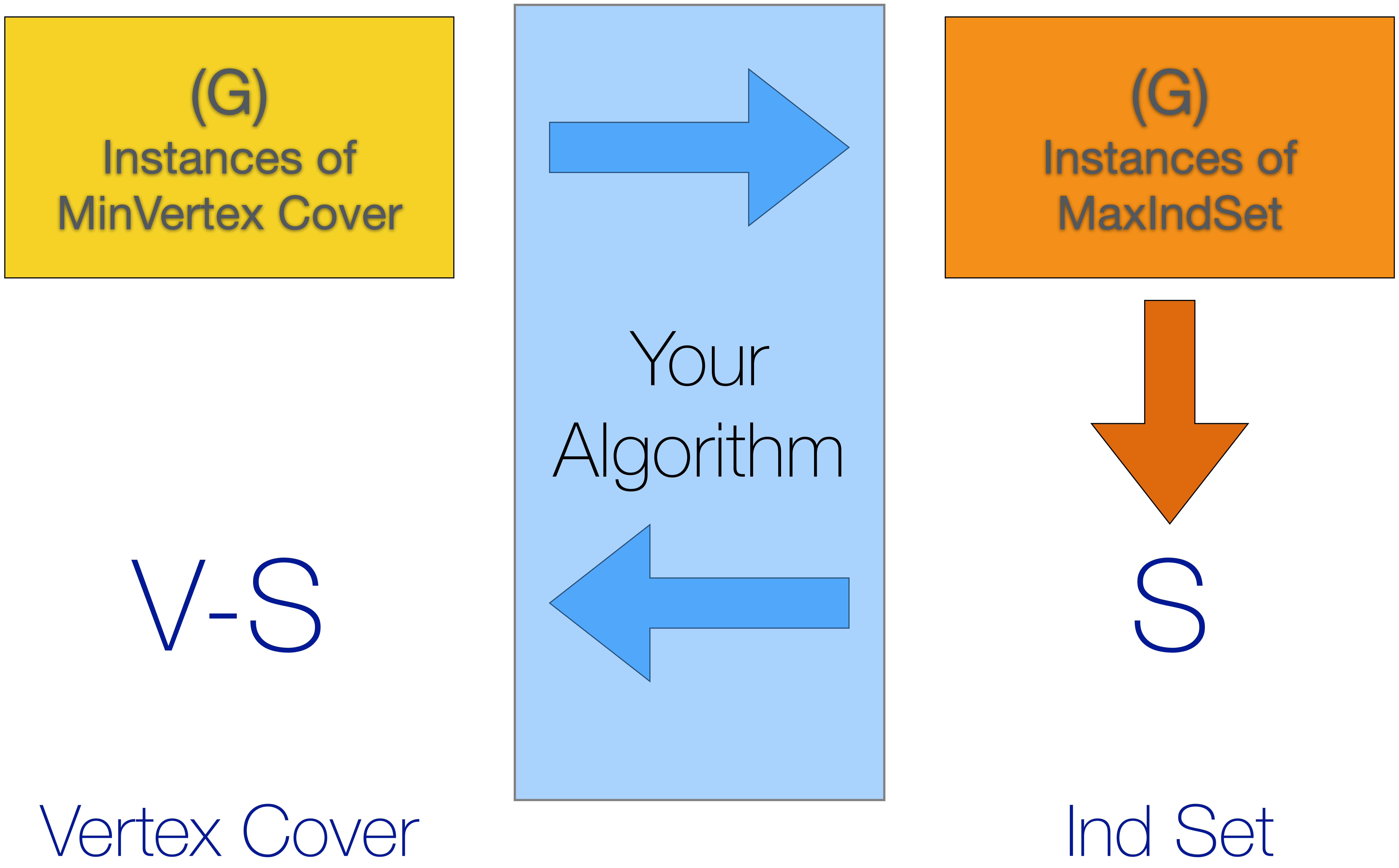
suppose S is an independent set.



THM: SET S IS AN INDEPENDENT SET OF G IFF V-S IS A VERTEX COVER.

SUPPOSE V-S IS A VC.





(G)
Instances of
MinVertex Cover
Your
Algorithm
(G)
Instances of
MaxIndSet
V-S
S
Vertex Cover
Ind Set

3sat problem

input:

output:

3sat problem

input: Boolean formula in 3CNF, i.e., a logical AND of clauses of the OR of 3 variables.

output:

3sat problem

input: Boolean formula in 3CNF, i.e., a logical AND of clauses of the OR of 3 variables.

output: An assignment A:V->{T,F} of variables that make the formula evaluate to True.

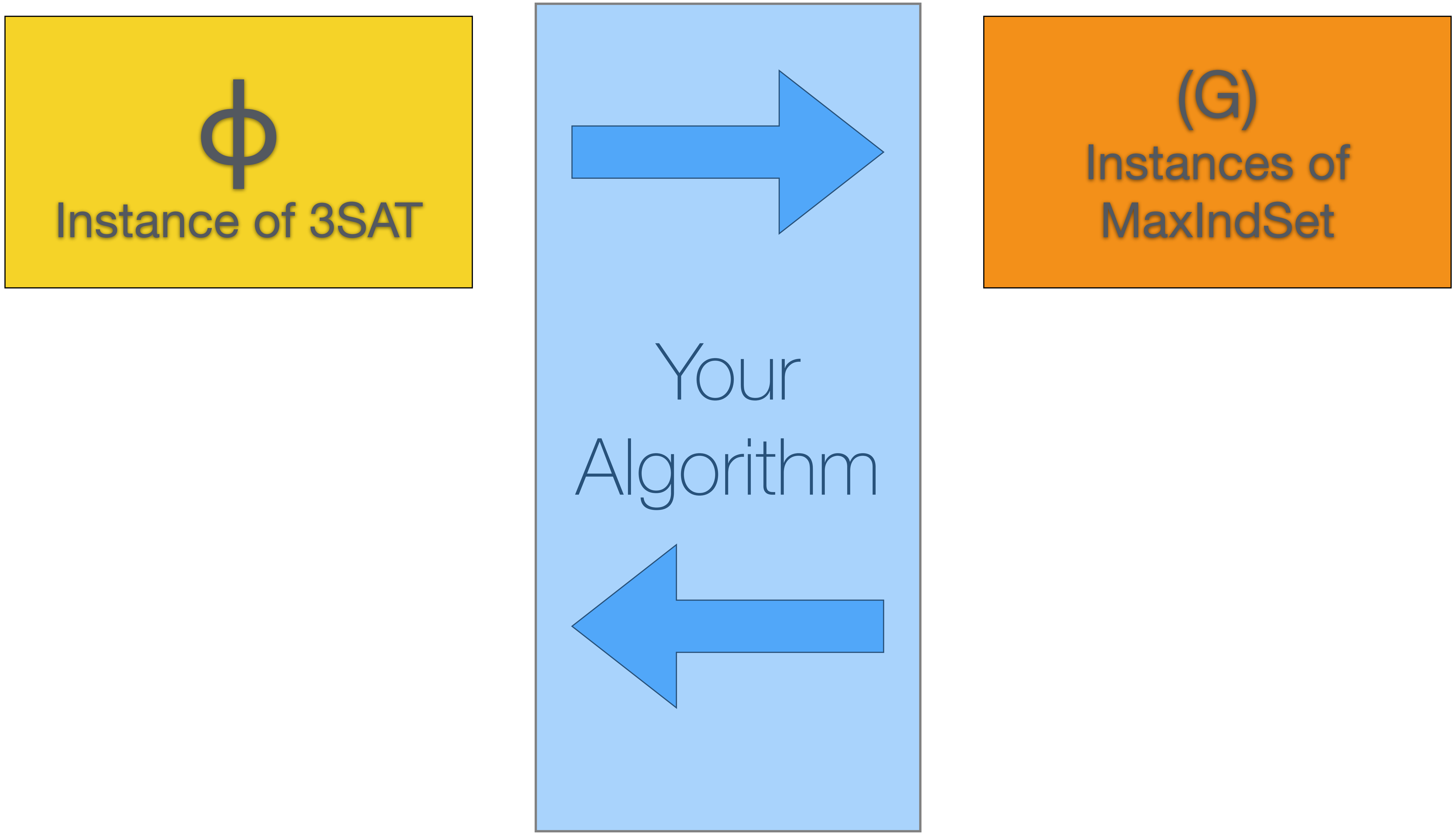
3sat example

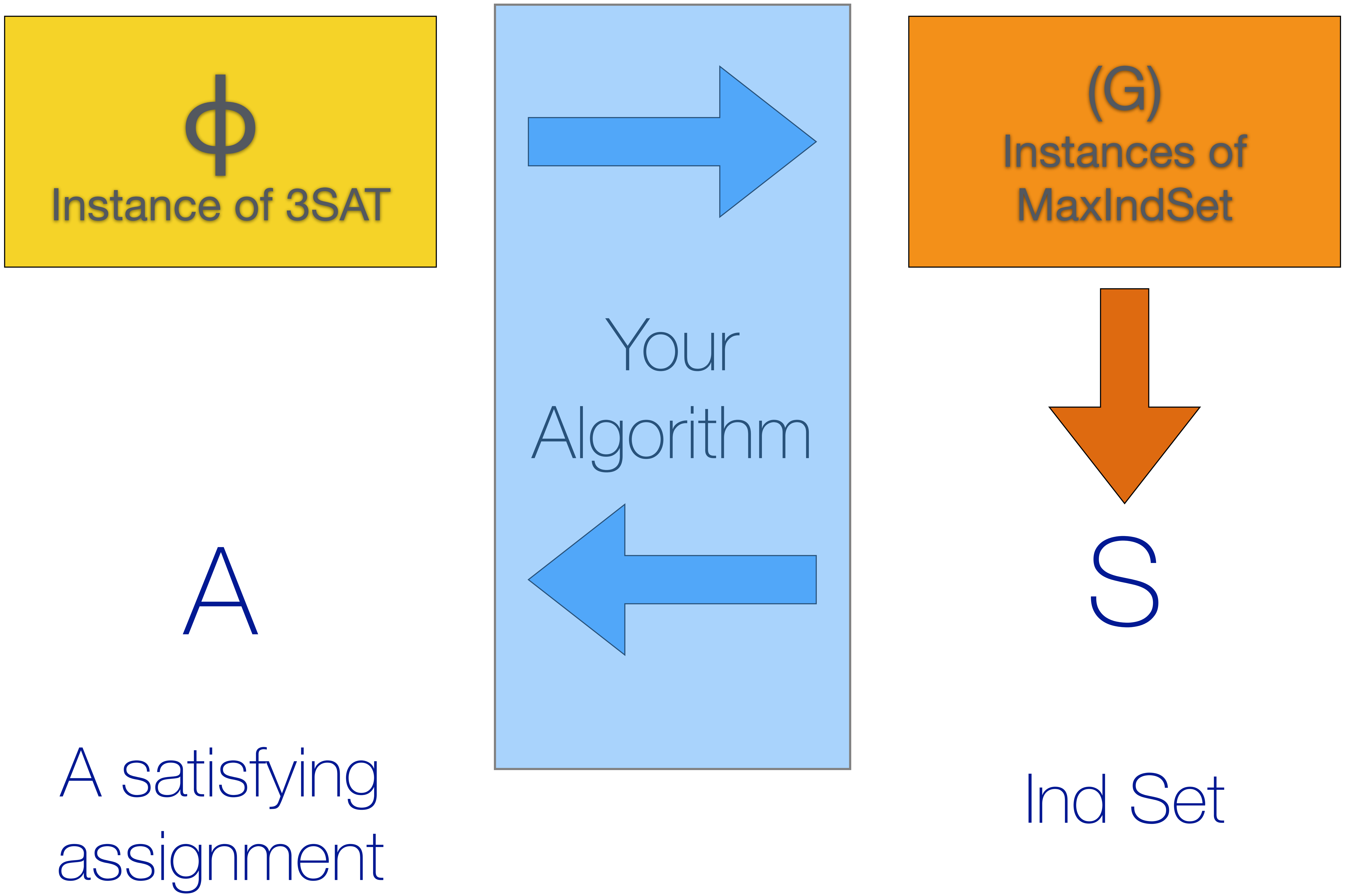
$$(x \vee y \vee z) \wedge (x \vee \overline{y} \vee y) \wedge (u \vee y \vee \overline{z}) \wedge (z \vee \overline{x} \vee u) \wedge (\overline{x} \vee \overline{y} \vee \overline{z})$$

$3\text{SAT} \leq_p \text{INDSET}$

$$(x \vee y \vee z) \wedge (x \vee \overline{y} \vee y) \wedge (u \vee y \vee \overline{z}) \wedge (z \vee \overline{x} \vee u) \wedge (\overline{x} \vee \overline{y} \vee \overline{z})$$

what must we do to?


ϕ
Instance of 3SAT
Your
Algorithm
(G)
Instances of
MaxIndSet


ϕ
Instance of 3SAT
Your
Algorithm
(G)
Instances of
MaxIndSet
A
A satisfying
assignment
S
Ind Set

These arguments often follow a common pattern.

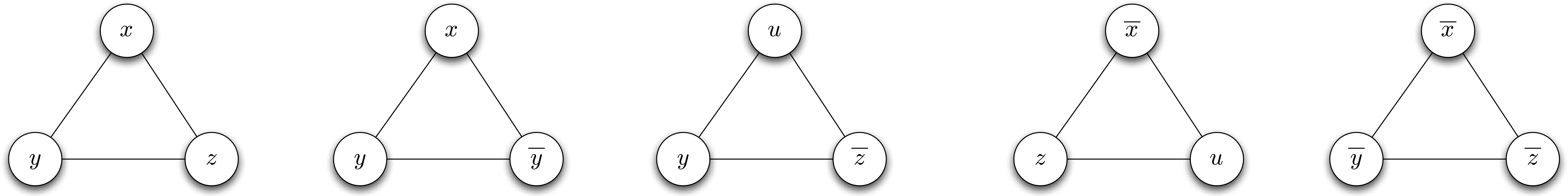
They involve a gadget that explains how to map aspects of one problem into another problem

$3\textsc{SAT} \leq_p \textsc{INDSET}$

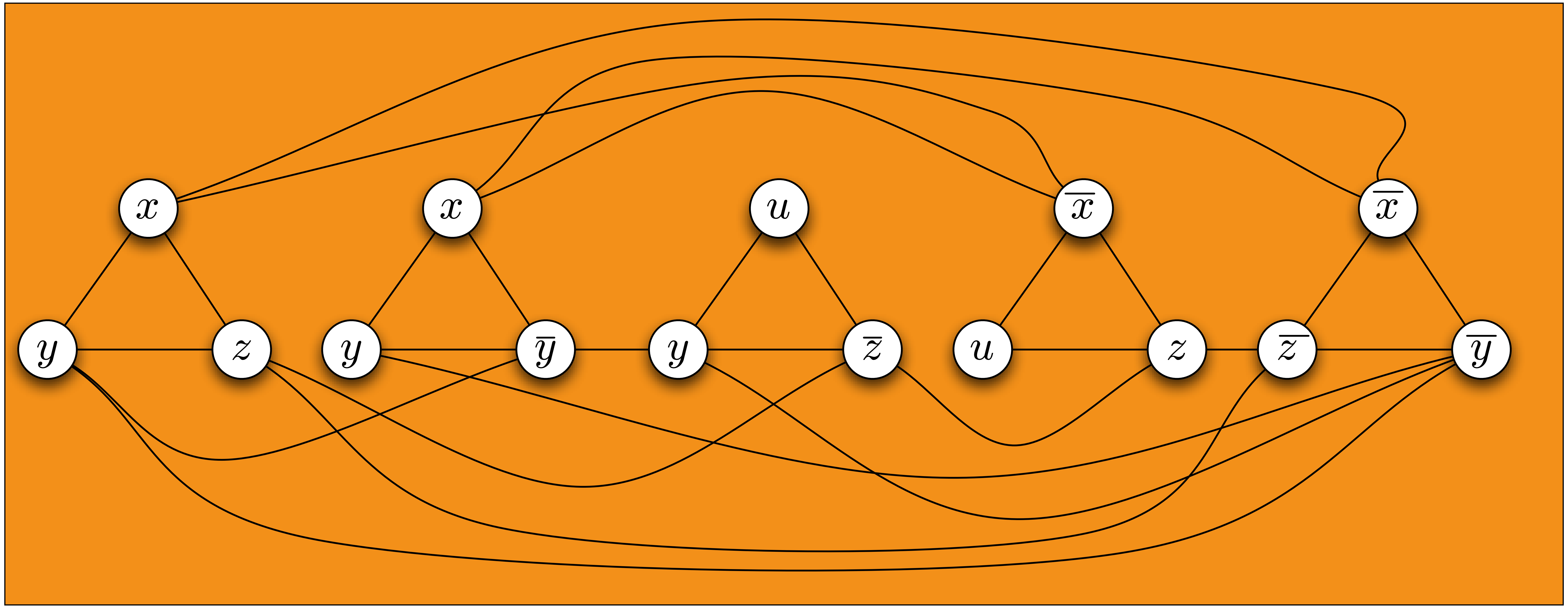
$$(x \vee y \vee z) \wedge (x \vee \overline{y} \vee y) \wedge (u \vee y \vee \overline{z}) \wedge (z \vee \overline{x} \vee u) \wedge (\overline{x} \vee \overline{y} \vee \overline{z})$$

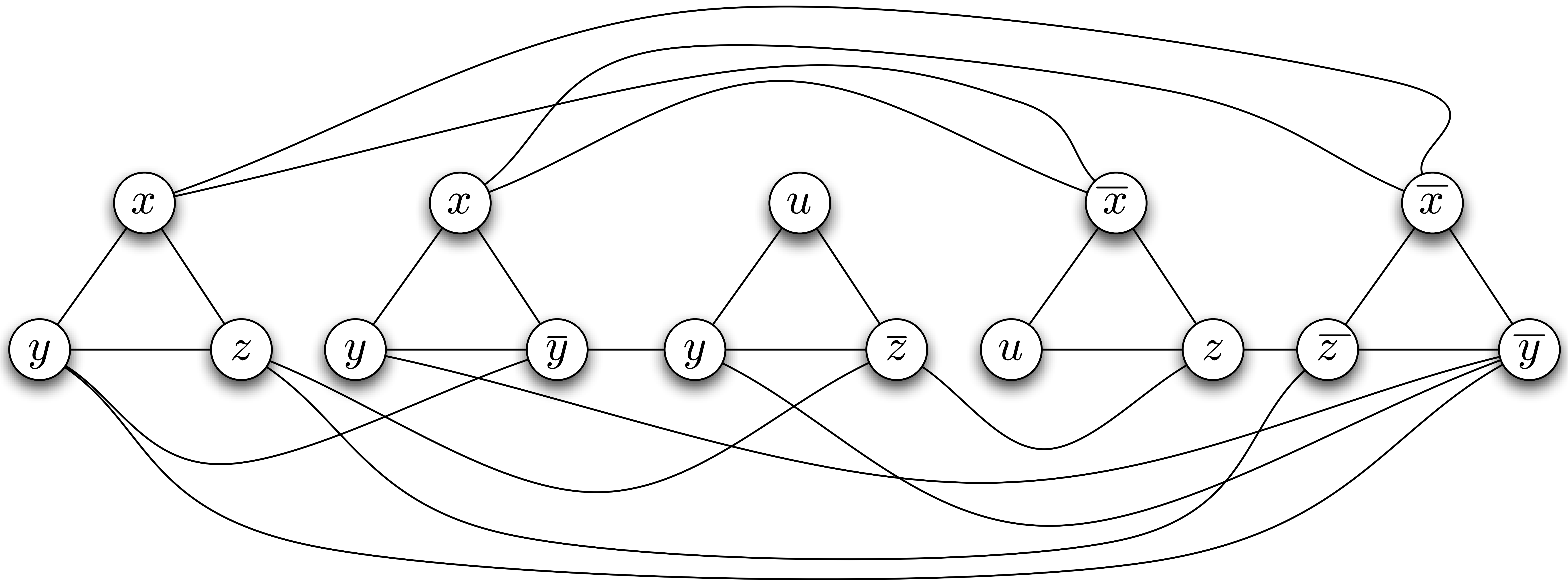
3SAT $\leq_p$ INDSET

$$(x \vee y \vee z) \wedge (x \vee \overline{y} \vee y) \wedge (u \vee y \vee \overline{z}) \wedge (z \vee \overline{x} \vee u) \wedge (\overline{x} \vee \overline{y} \vee \overline{z})$$

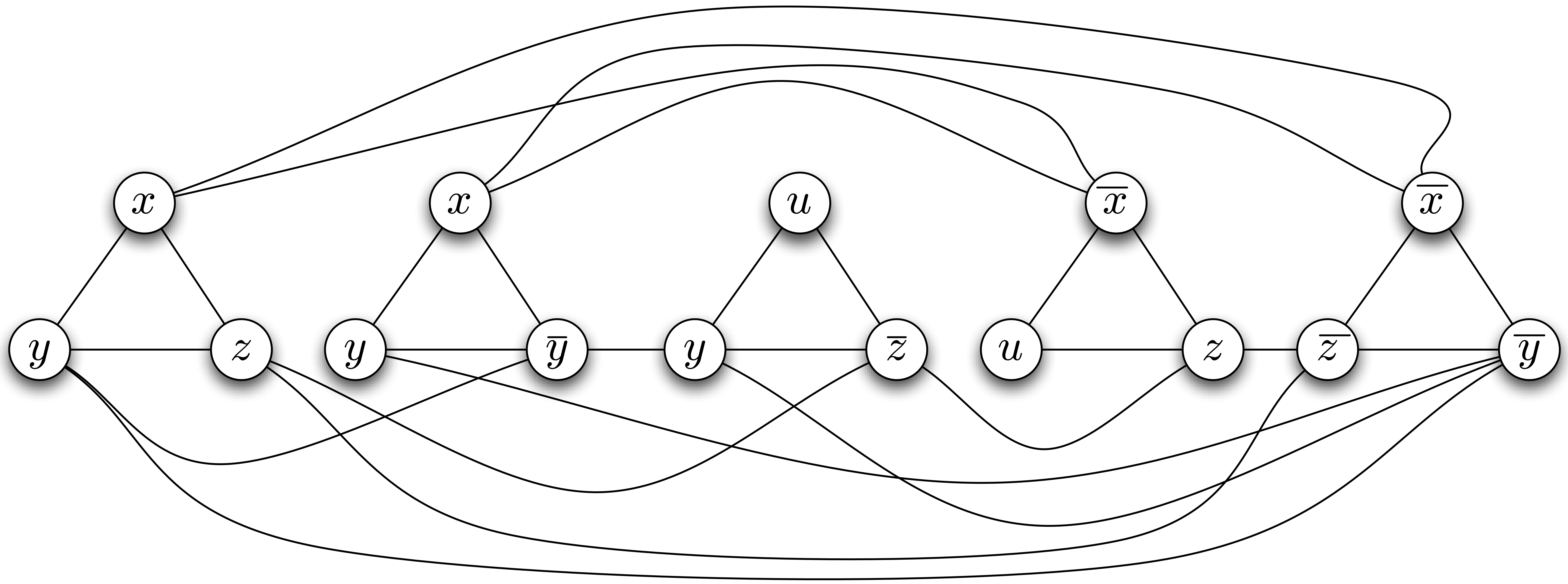


$$(x \vee y \vee z) \wedge (x \vee \overline{y} \vee y) \wedge (u \vee y \vee \overline{z}) \wedge (z \vee \overline{x} \vee u) \wedge (\overline{x} \vee \overline{y} \vee \overline{z})$$



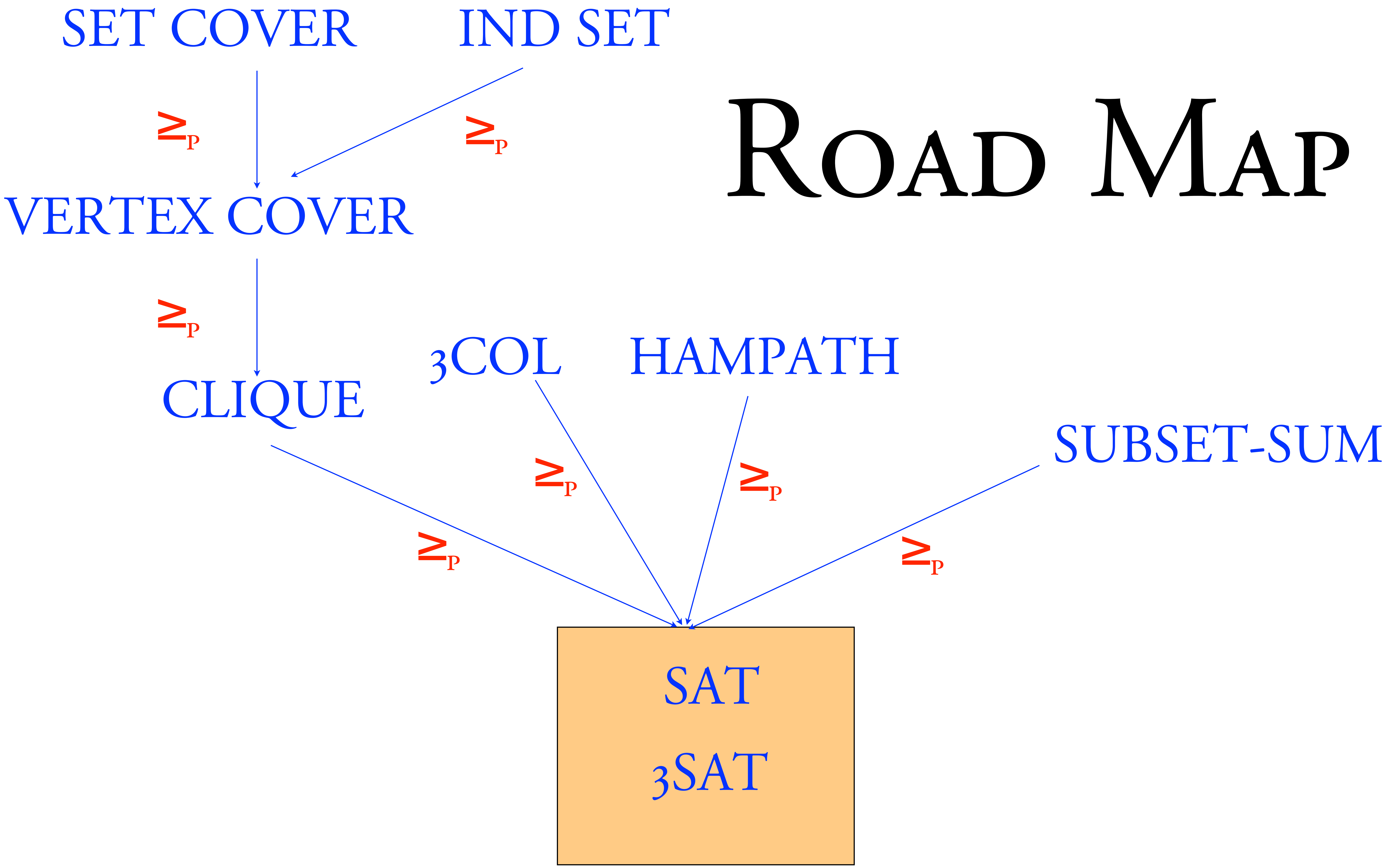


$\phi \in \mathrm{SAT} \implies$

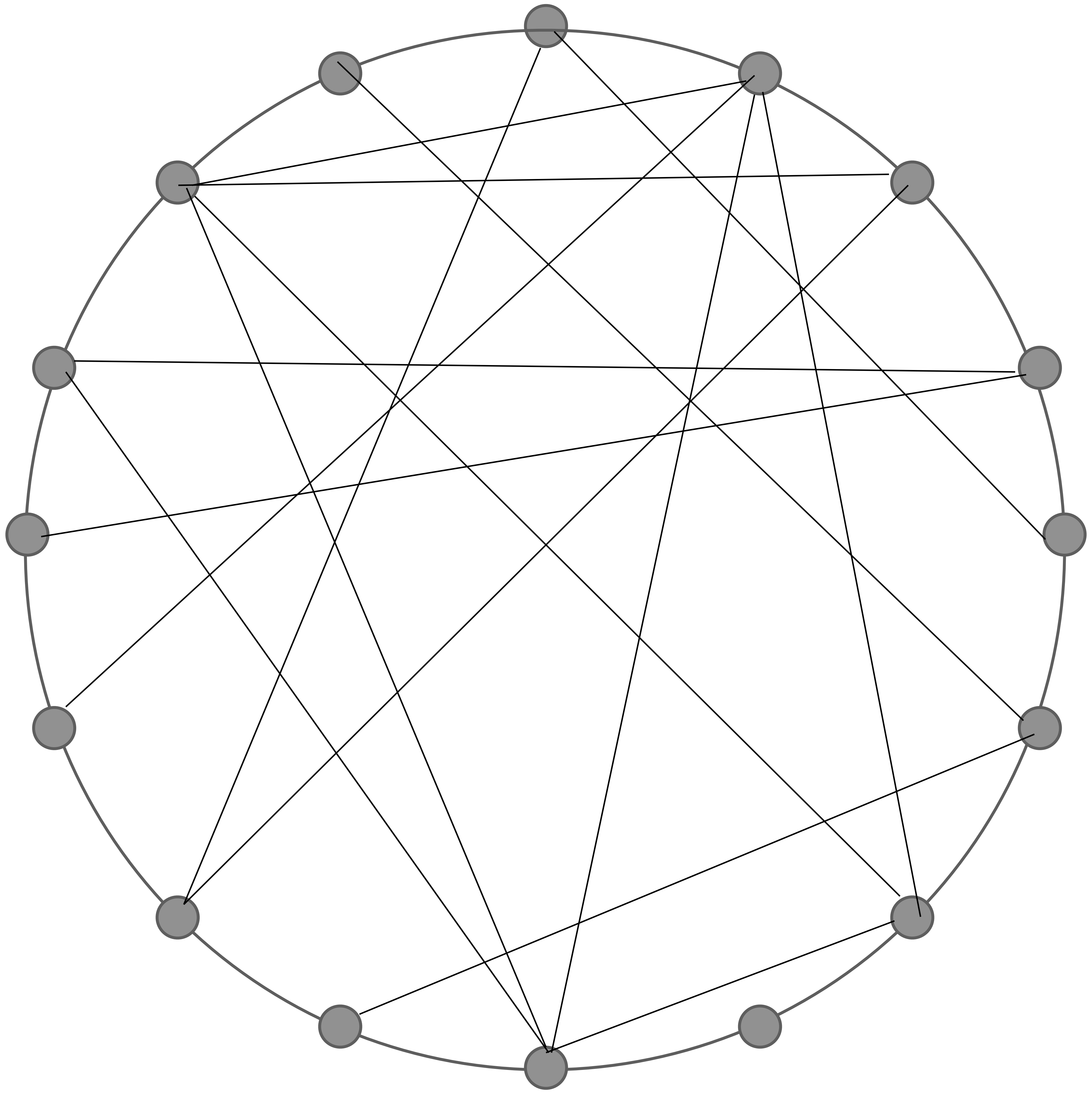


$(\mathrm{G}, \mathrm{k}) \in \text{INDSET} \implies$

Road Map

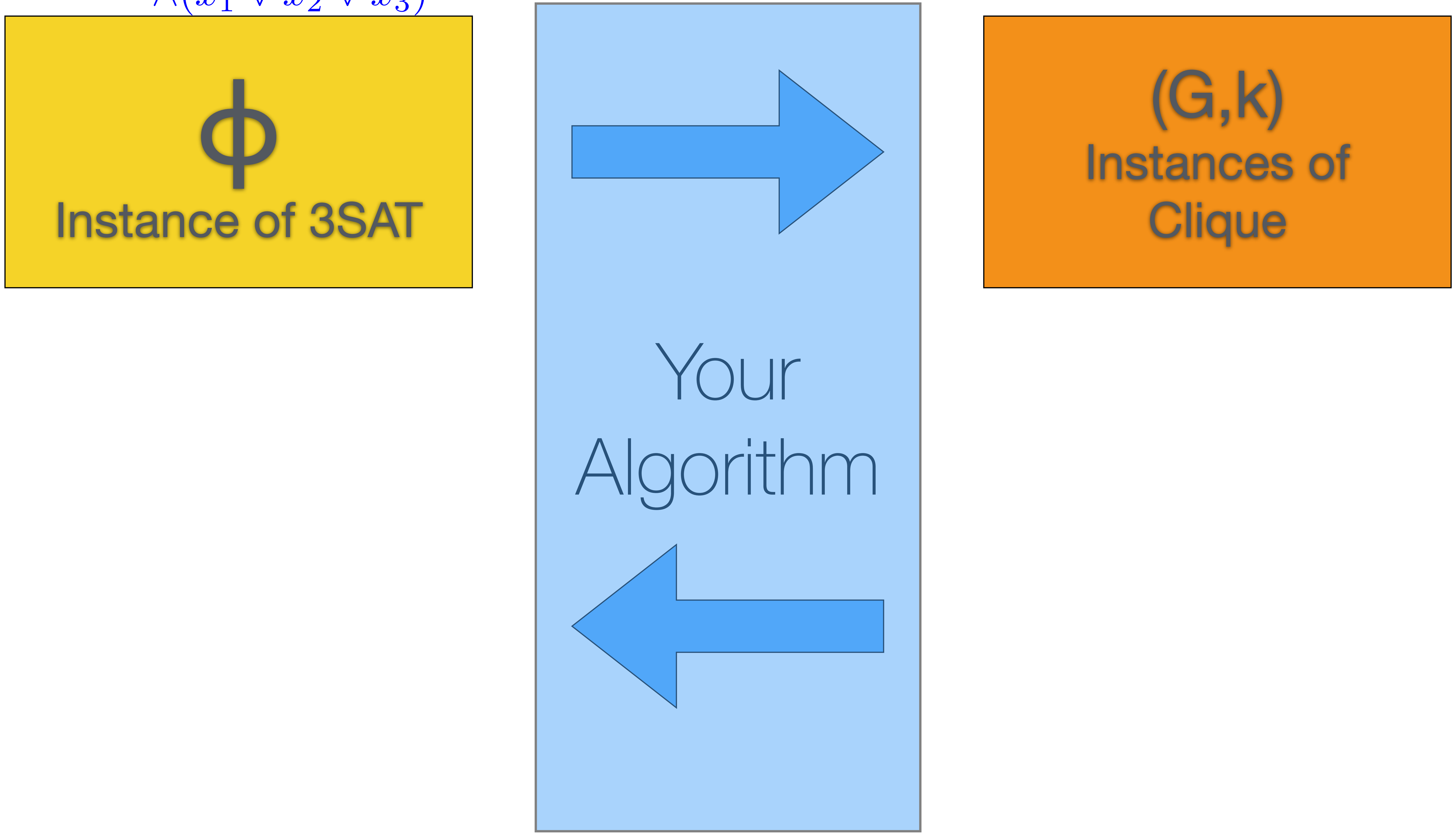

SET COVER
IND SET
VERTEX COVER
CLIQUE
3COL
HAMPATH
SUBSET-SUM
SAT
3SAT
$\geq_P$

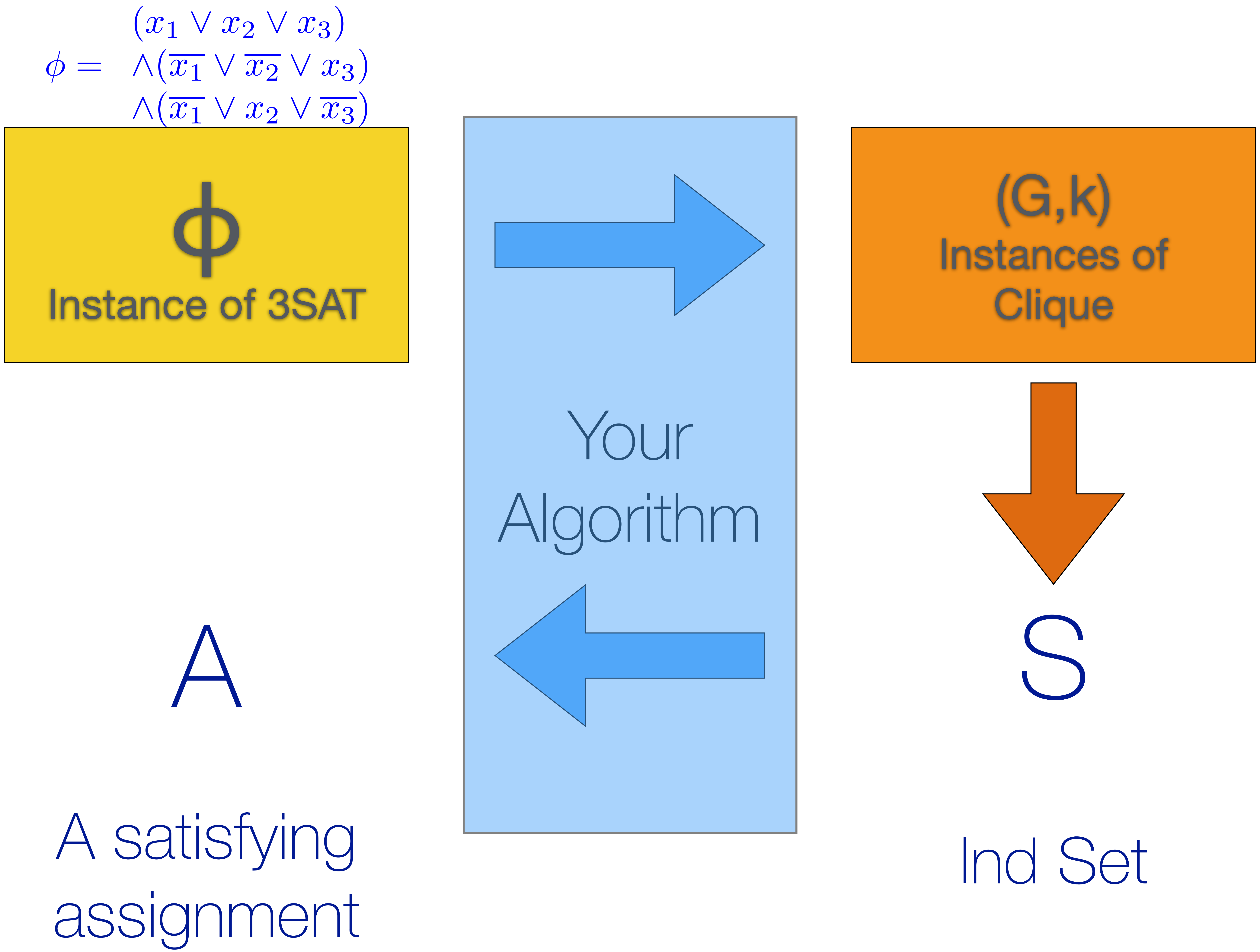
clique



clique = {

$$\phi = \begin{array}{l} (x_1 \vee x_2 \vee x_3) \\ \wedge(\overline{x_1} \vee \overline{x_2} \vee x_3) \\ \wedge(\overline{x_1} \vee x_2 \vee \overline{x_3}) \end{array}$$




$\phi = (x_1 \vee x_2 \vee x_3) \wedge(\overline{x_1} \vee \overline{x_2} \vee x_3) \wedge(\overline{x_1} \vee x_2 \vee \overline{x_3})$
ϕ
Instance of 3SAT
Your Algorithm
(G,k)
Instances of Clique
A
A satisfying assignment
S
Ind Set

CLIQUE

FORMULA

$$\phi = \begin{array}{l}(x_1 \vee x_2 \vee x_3) \\ \wedge(\overline{x_1} \vee \overline{x_2} \vee x_3) \\ \wedge(\overline{x_1} \vee x_2 \vee \overline{x_3})\end{array}$$

F

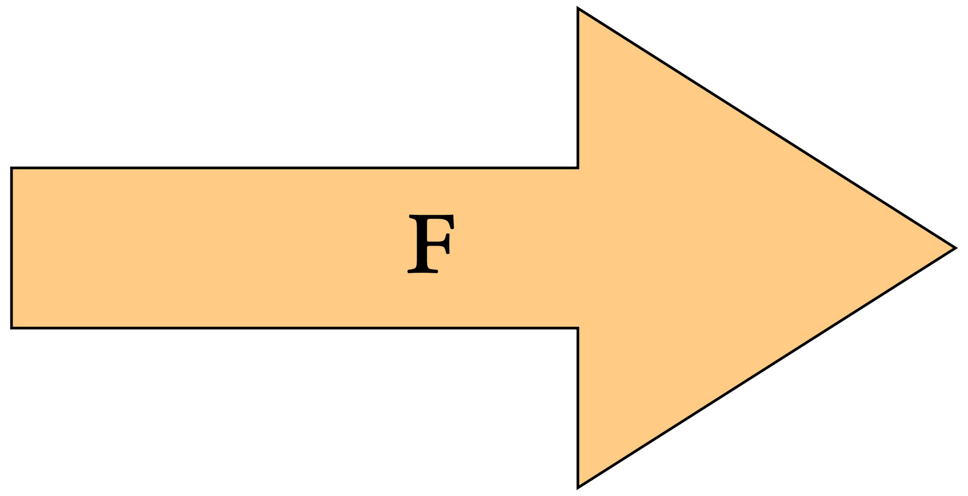
GRAPH, K

K = # CLAUSES

CLIQUE

FORMULA

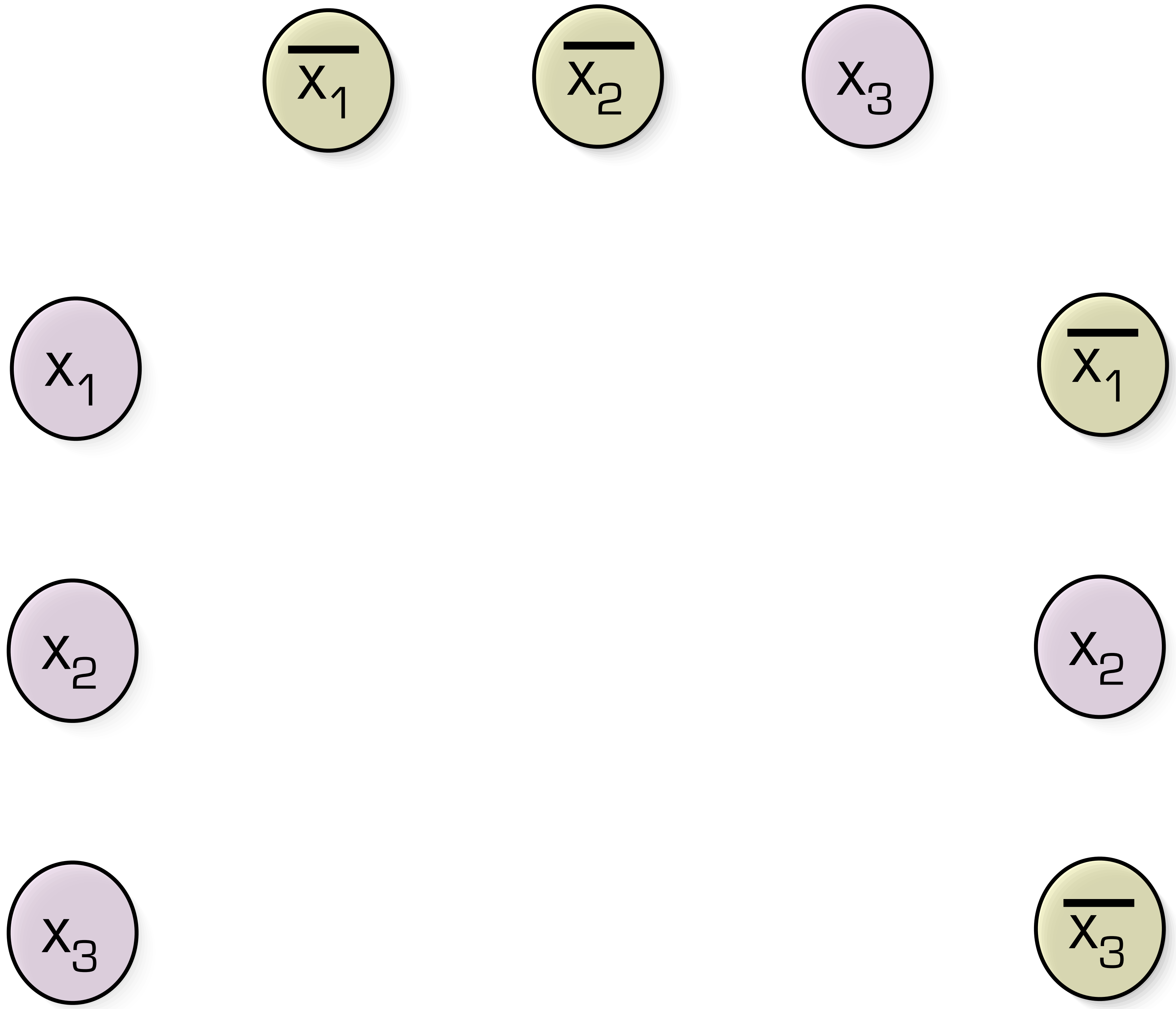
$$\phi = \begin{array}{l} (x_1 \vee x_2 \vee x_3) \\ \wedge(\overline{x_1} \vee \overline{x_2} \vee x_3) \\ \wedge(\overline{x_1} \vee x_2 \vee \overline{x_3}) \end{array}$$



GRAPH, K

K = # CLAUSES

CREATE 3 NODES/CLAUSE



CLIQUE

FORMULA

$$\phi = \begin{array}{l} (x_1 \vee x_2 \vee x_3) \\ \wedge(\overline{x_1} \vee \overline{x_2} \vee x_3) \\ \wedge(\overline{x_1} \vee x_2 \vee \overline{x_3}) \end{array}$$

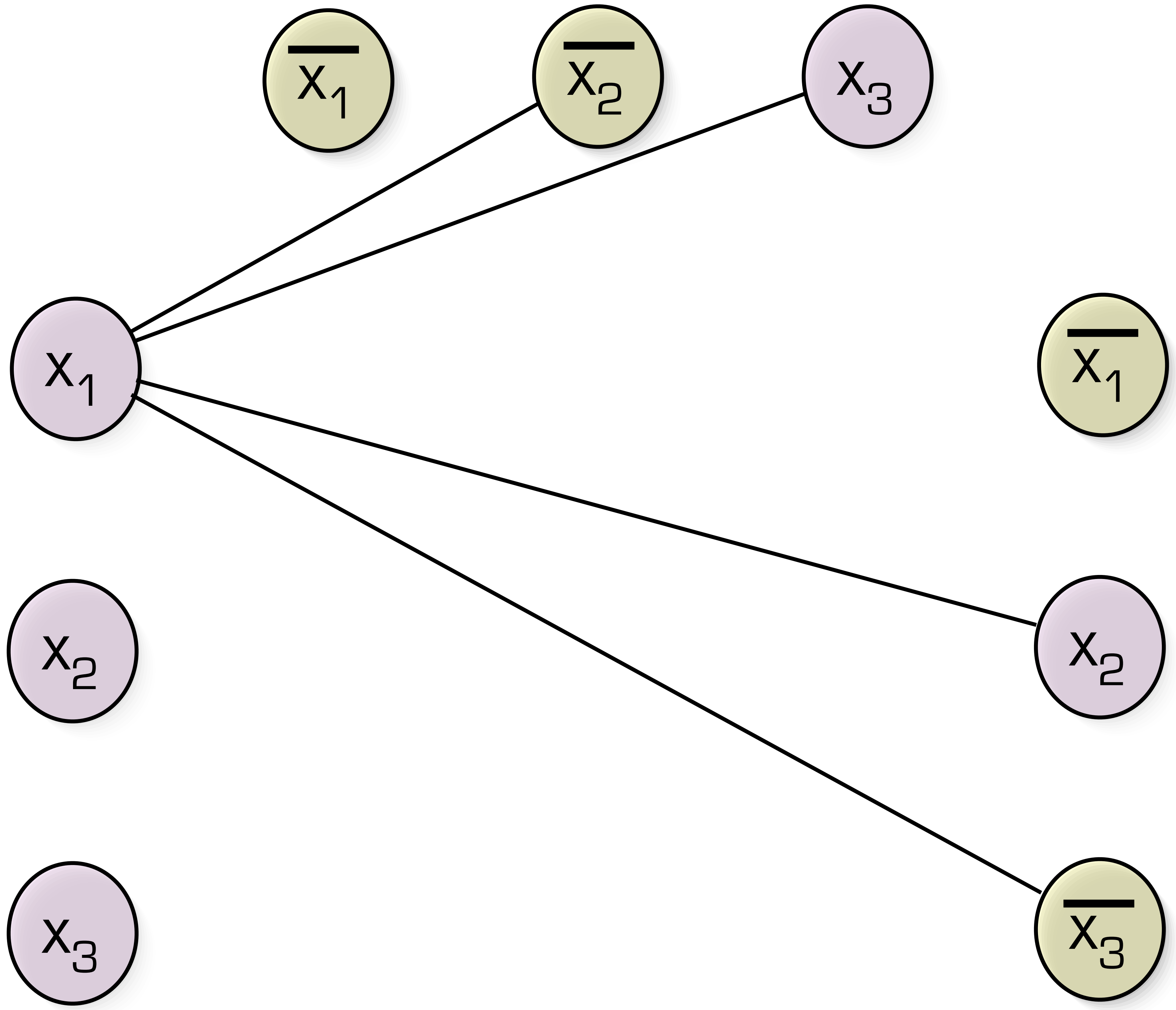
F

GRAPH, K

K = # CLAUSES

Create 3 nodes/clause

Connect nodes to
"non-opposites"



CLIQUE

FORMULA

$$\phi = \begin{array}{l}(x_1 \vee x_2 \vee x_3) \\ \wedge(\overline{x_1} \vee \overline{x_2} \vee x_3) \\ \wedge(\overline{x_1} \vee x_2 \vee \overline{x_3})\end{array}$$

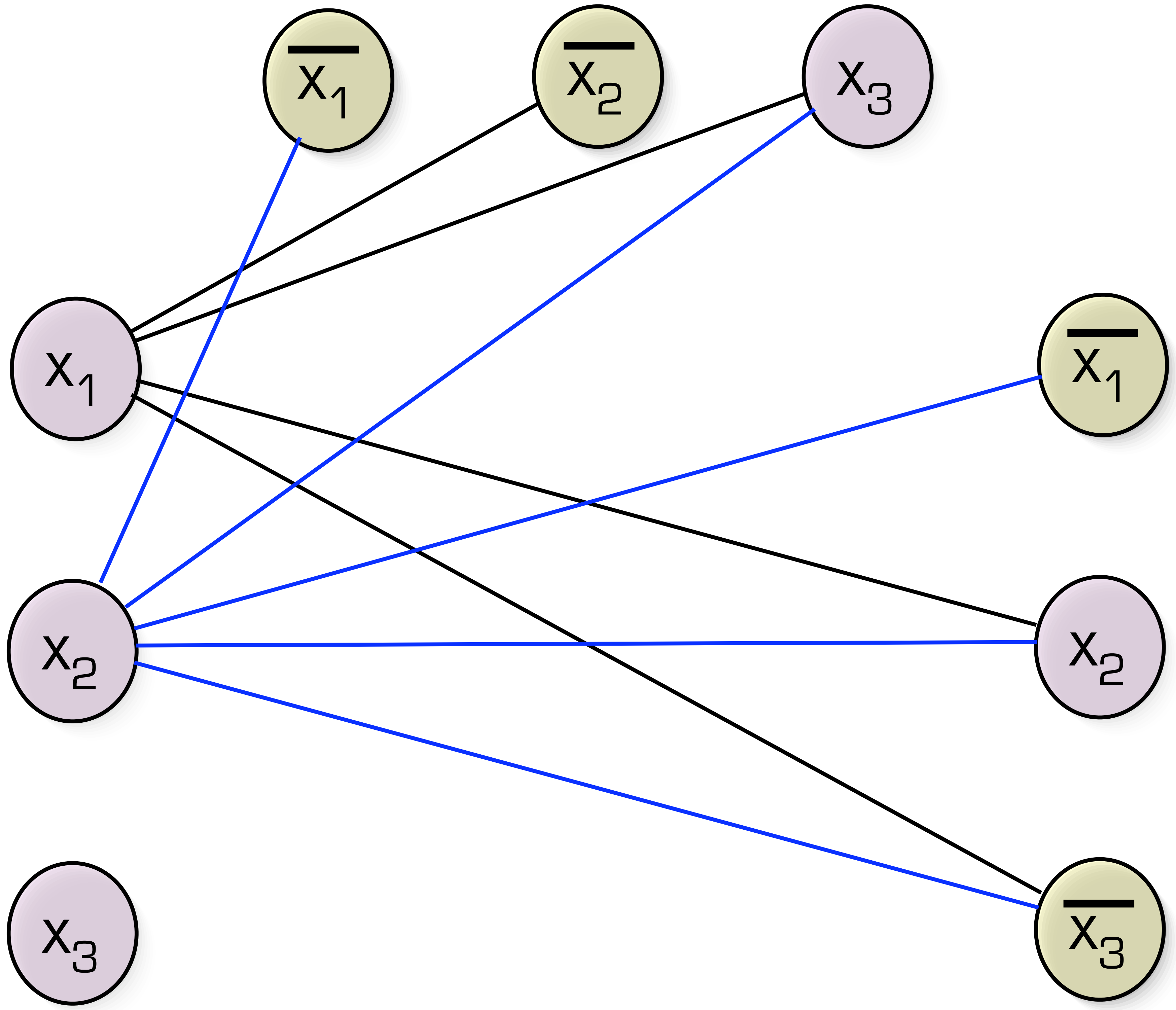
F

GRAPH, K

K = # CLAUSES

Create 3 nodes/clause

Connect nodes to
"non-opposites"



CLIQUE

FORMULA

$$\phi = \begin{array}{l}(x_1 \vee x_2 \vee x_3) \\ \wedge(\overline{x_1} \vee \overline{x_2} \vee x_3) \\ \wedge(\overline{x_1} \vee x_2 \vee \overline{x_3})\end{array}$$

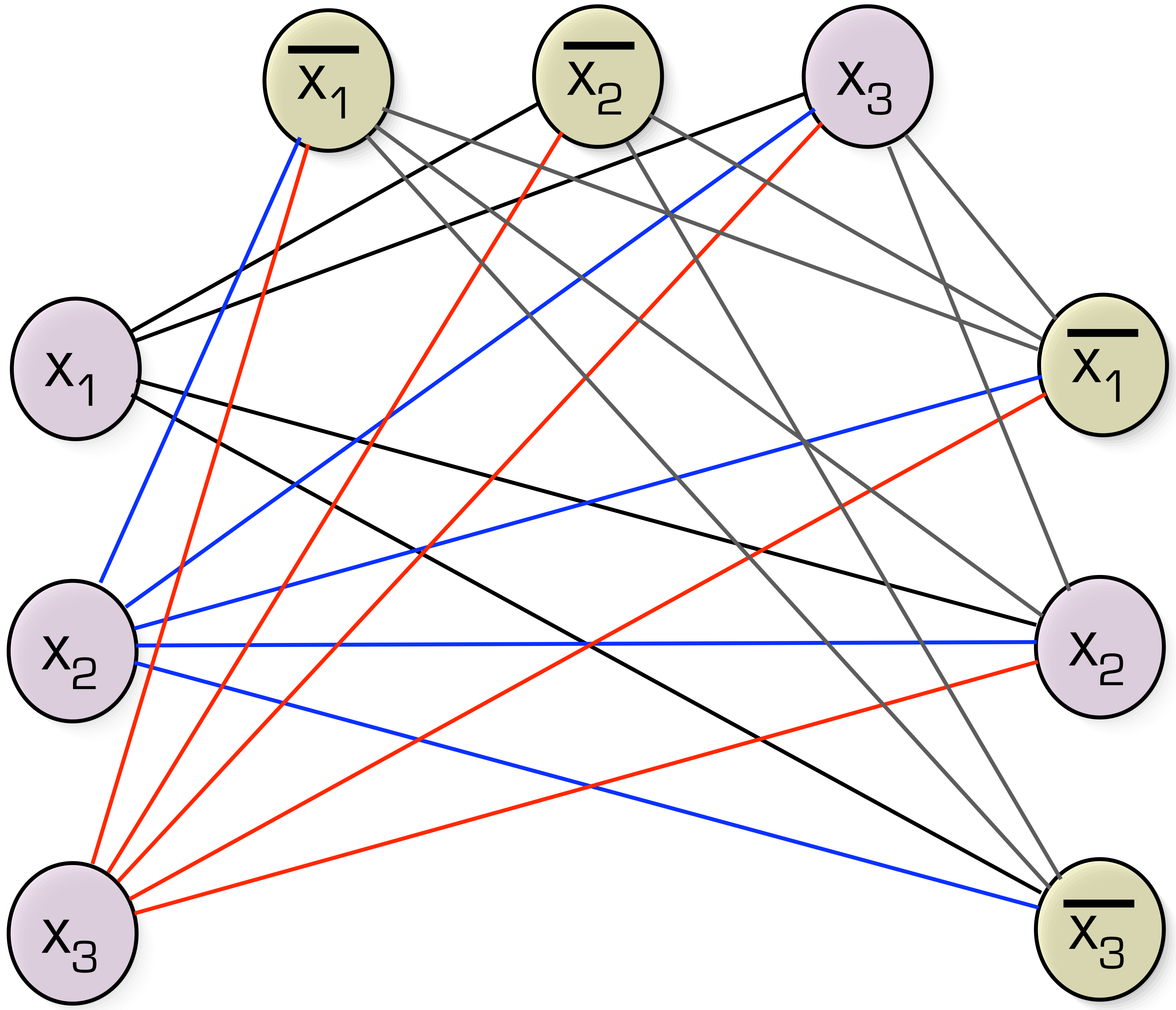
F

GRAPH, K

K = # CLAUSES

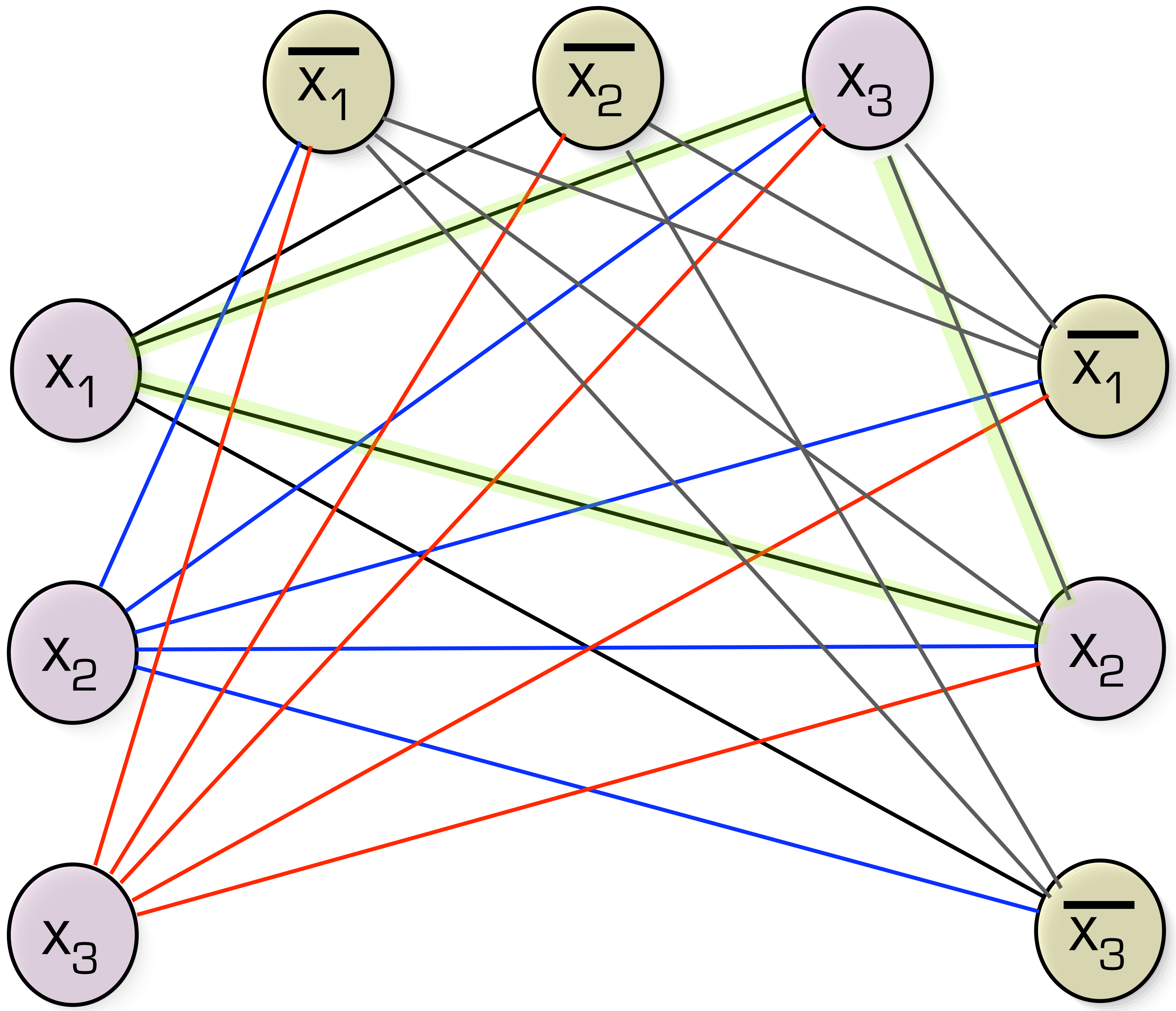
Create 3 nodes/clause

Connect nodes to
"non-opposites"



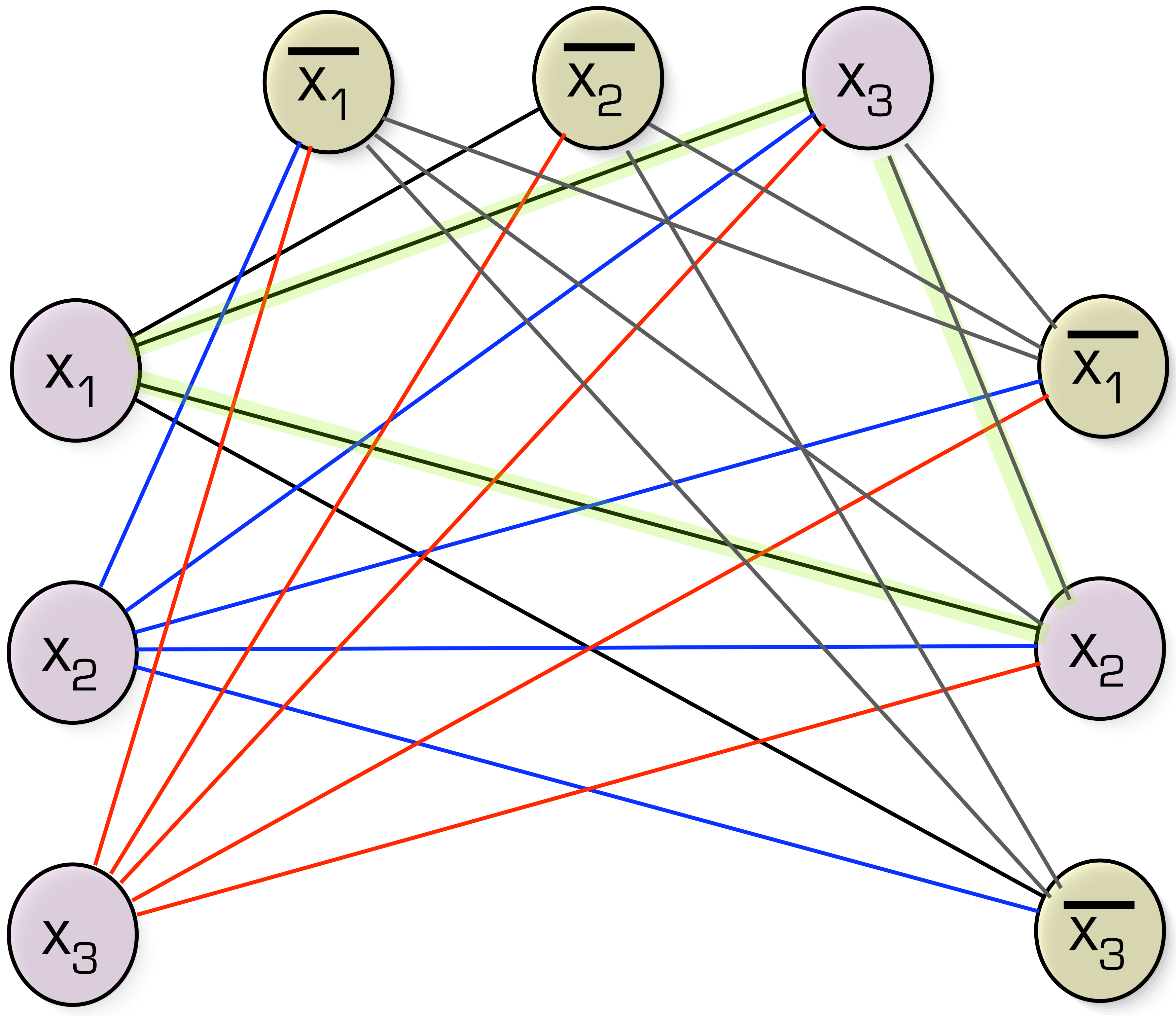
CLIQUE

$$\phi = \begin{array}{l}(x_1 \vee x_2 \vee x_3) \\ \wedge(\overline{x_1} \vee \overline{x_2} \vee x_3) \\ \wedge(\overline{x_1} \vee x_2 \vee \overline{x_3})\end{array}$$



CLIQUE

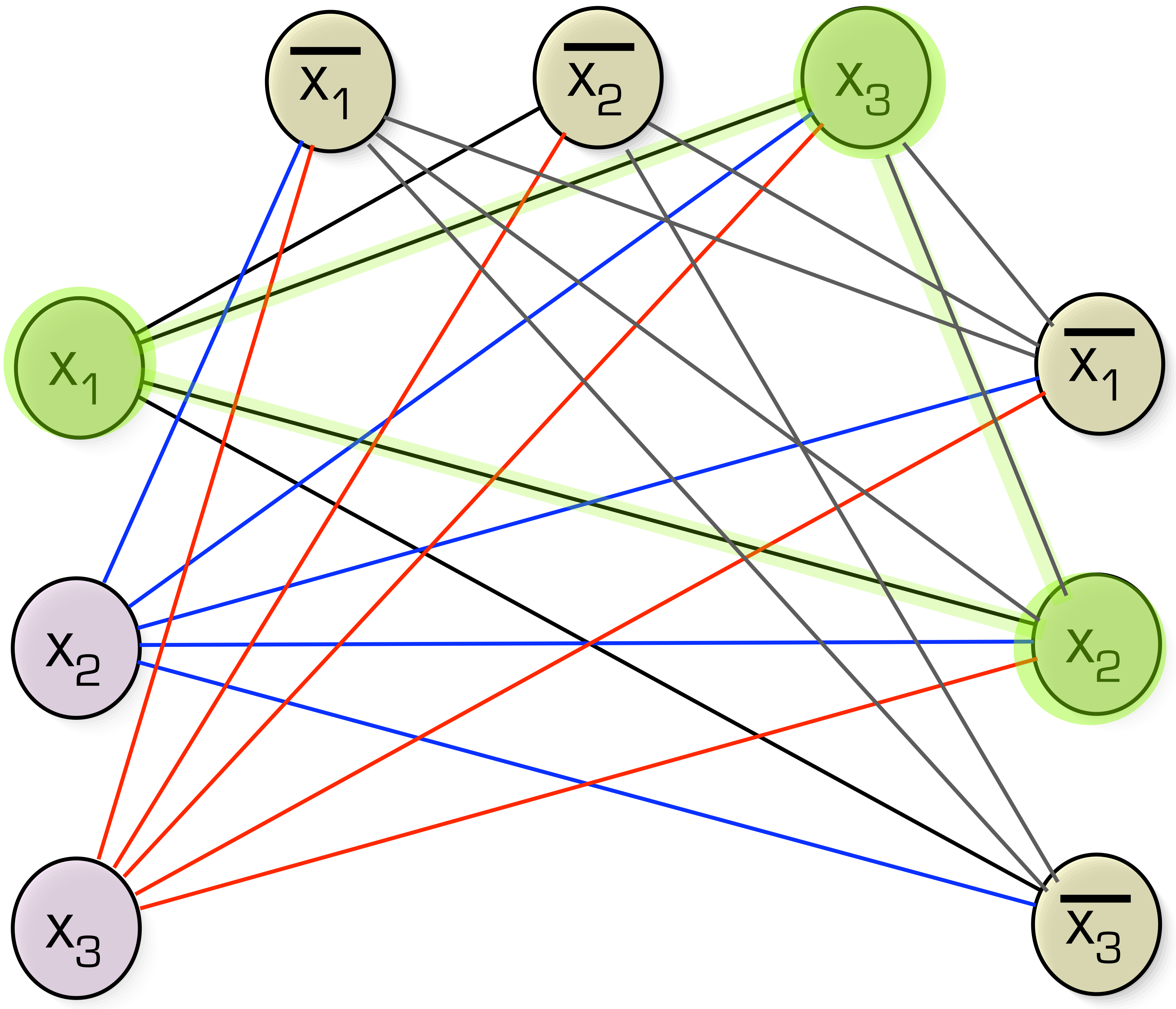
$$\phi = \begin{array}{l} (x_1 \vee x_2 \vee x_3) \\ \wedge(\overline{x_1} \vee \overline{x_2} \vee x_3) \\ \wedge(\overline{x_1} \vee x_2 \vee \overline{x_3}) \end{array}$$



Satisfying assignment = 1 var/clause

CLIQUE

$$\phi = \begin{array}{l} (x_1 \vee x_2 \vee x_3) \\ \wedge(\overline{x_1} \vee \overline{x_2} \vee x_3) \\ \wedge(\overline{x_1} \vee x_2 \vee \overline{x_3}) \end{array}$$

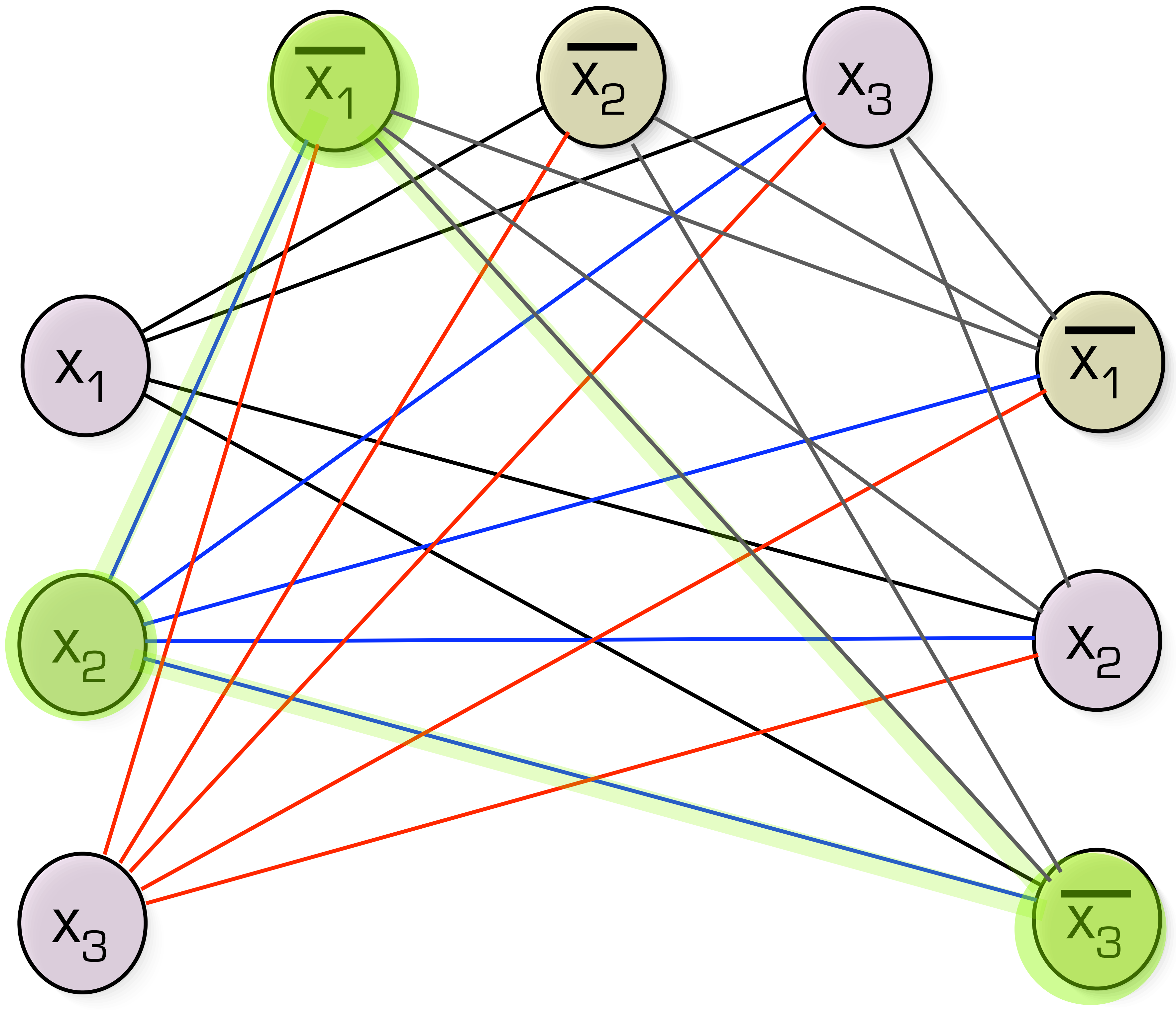


Satisfying assignment = 1 var/clause

k "non-opposite" connected nodes

CLIQUE

$$\phi = \begin{array}{l} (x_1 \vee x_2 \vee x_3) \\ \wedge(\overline{x_1} \vee \overline{x_2} \vee x_3) \\ \wedge(\overline{x_1} \vee x_2 \vee \overline{x_3}) \end{array}$$

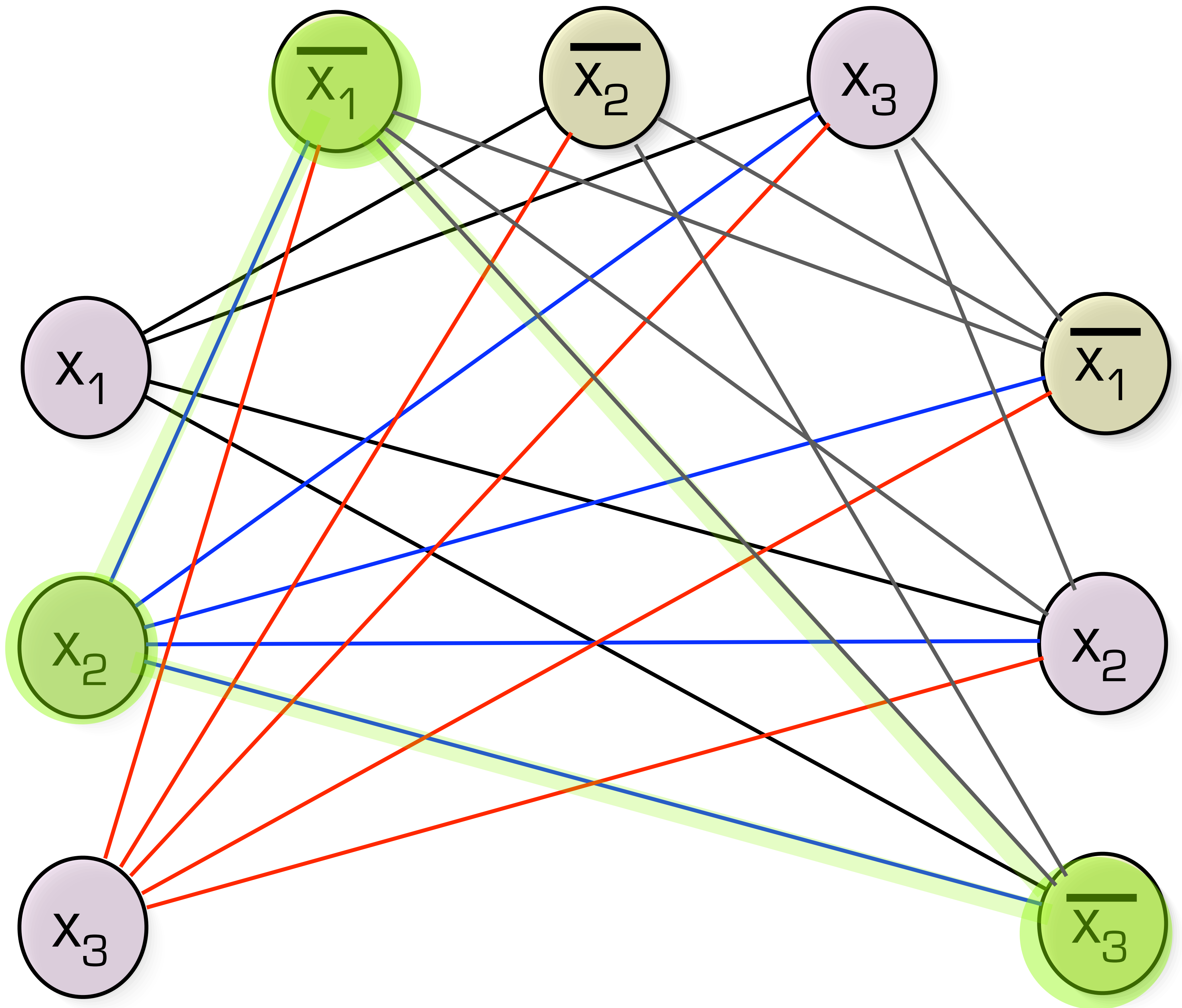


K-CLIQUE

1 NODE/CLAUSE IS TRUE

CLIQUE

$$\phi = \begin{array}{l} (x_1 \vee x_2 \vee x_3) \\ \wedge(\overline{x_1} \vee \overline{x_2} \vee x_3) \\ \wedge(\overline{x_1} \vee x_2 \vee \overline{x_3}) \end{array}$$



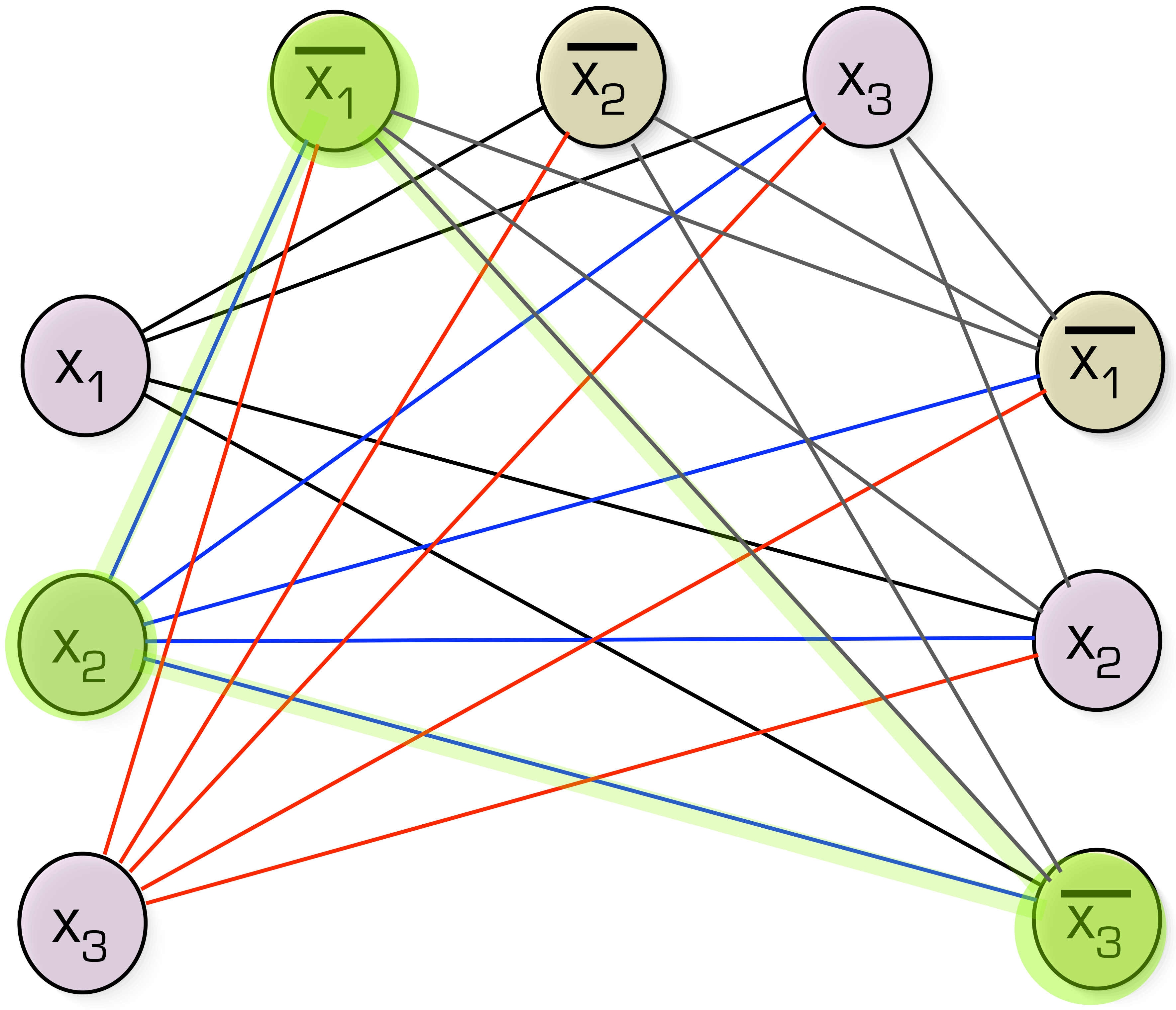
K-CLIQUE

1 NODE/CLAUSE IS TRUE

SATISFYING ASSIGNMENT

CLIQUE

$$\phi = \begin{array}{l}(x_1 \vee x_2 \vee x_3) \\ \wedge(\overline{x_1} \vee \overline{x_2} \vee x_3) \\ \wedge(\overline{x_1} \vee x_2 \vee \overline{x_3})\end{array}$$



$$\phi \in SAT \quad \Leftrightarrow \quad f(\phi) \in CLIQUE$$

Theory of NP

Languages

DEF OF NP

a language L belongs to the class NP iff
there exists a polynomial time algorithm A
and a constant c such that

$$L = \{x \in \{0,1\}^* \mid \exists y \in \{0,1\}^{|x|^c} \text{ s.t. } A(x,y) = 1\}$$

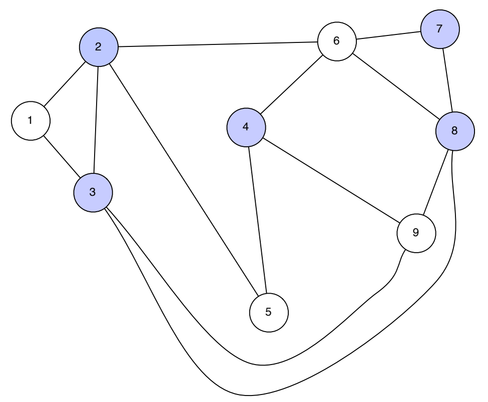
NP-Completeness

A language L is NP-Complete if

1. $L \in NP$
2. $\forall A \in NP,\ A \leq_P L$

"L is among the hardest NP problems"

WHY IS VC IN NP?



vertexcover(G,k)

COOK-LEVIN THEOREM

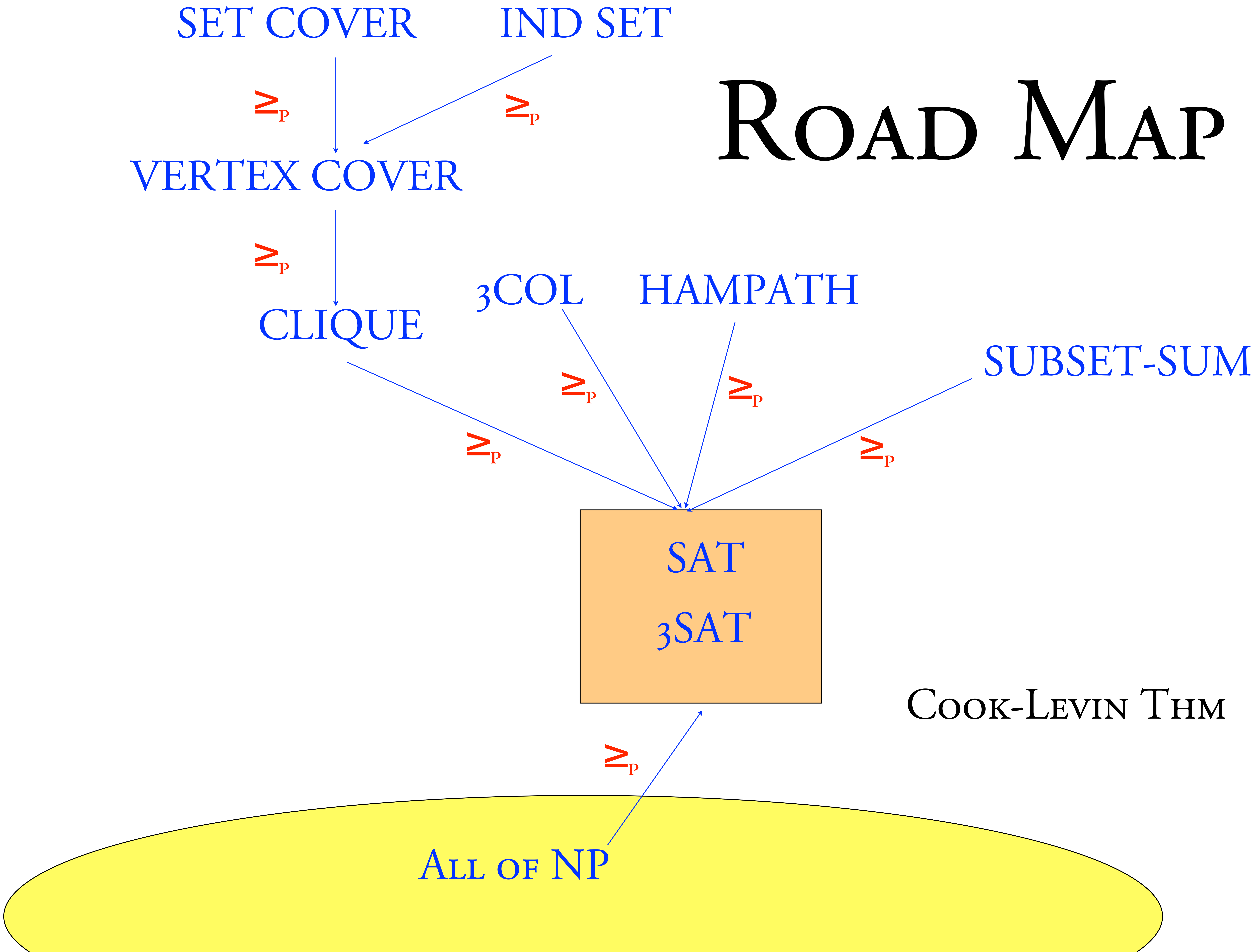
WHAT IS THE HARDEST PROBLEM IN NP?

Cook-Levin theorem

$$\forall L \in \mathrm{NP}$$

$$L \leq_f \mathrm{3SAT}$$

Road Map


SET COVER
IND SET
$\geq_P$
$\geq_P$
VERTEX COVER
$\geq_P$
CLIQUE
3COL
HAMPATH
SUBSET-SUM
$\geq_P$
$\geq_P$
$\geq_P$
$\geq_P$
SAT
3SAT
Cook-Levin Thm
$\geq_P$
All of NP