5800 Reductions

apr5/apr7 2022
shelat

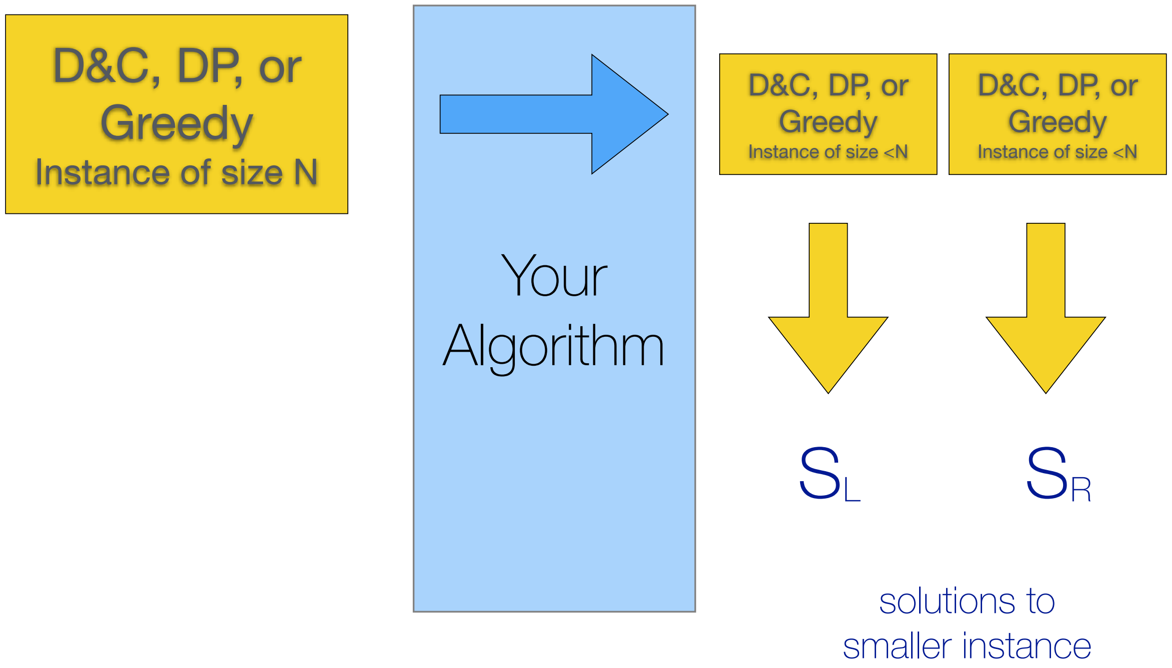
WE HAVE BEEN SOLVING PROBLEM A BY SOLVING SMALLER VERSIONS OF PROBLEM A

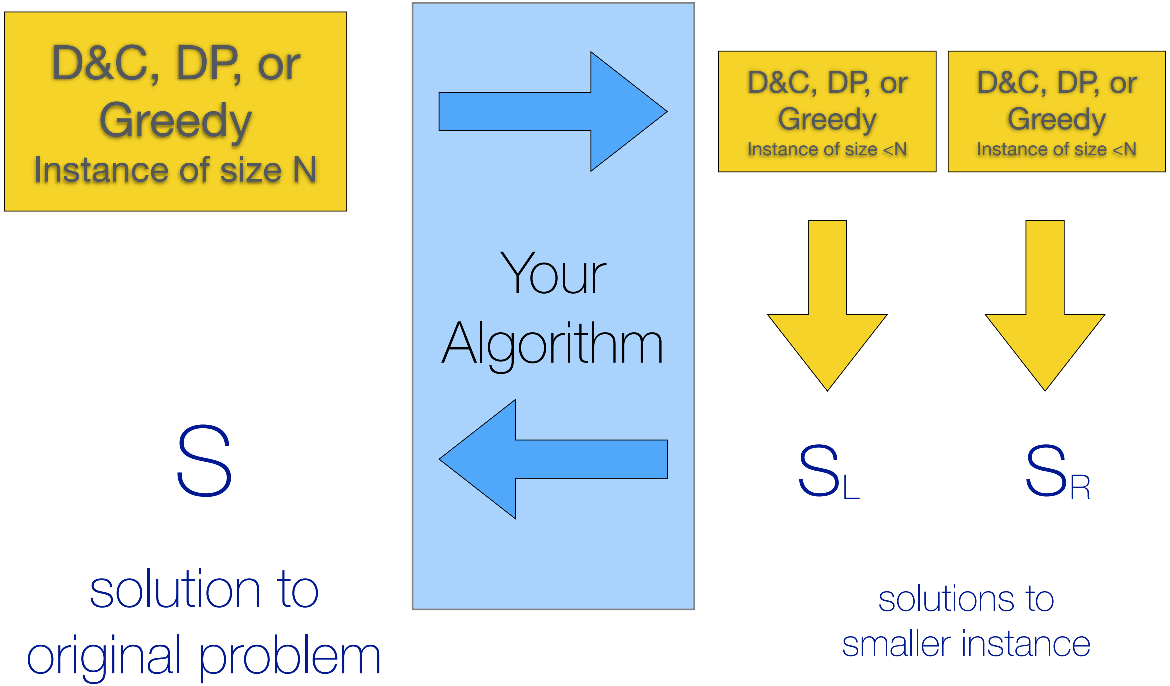
MORE GENERAL IDEA:

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

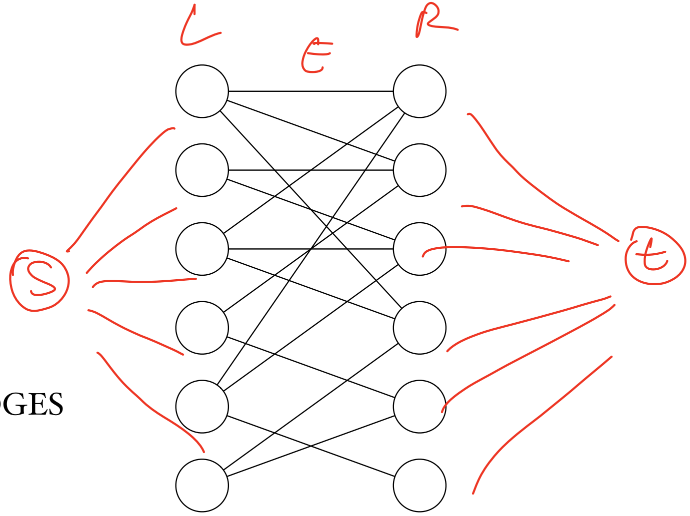
Bipartite Matching Algorithm

BP(L,R,E)

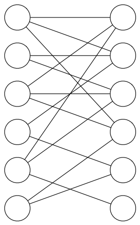
1. MAKE NEW G'
FROM INPUT G.

2. RUN FF ON G'

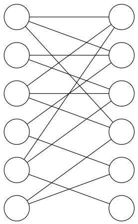
3. OUTPUT ALL MIDDLE EDGES
WITH FLOW F(E)=1.



Bipartite Matching Instance

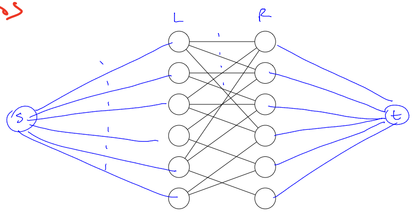


Bipartite Matching Instance



Max flow Instance

how to transform



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 $f(s, x) = 1$

$f(y, t) = 1$

⇒ flow f satisfies ① capacity constraint

② flow constraint

$\text{INFLOW}(x) = \text{OUTFLOW}(x)$

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

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

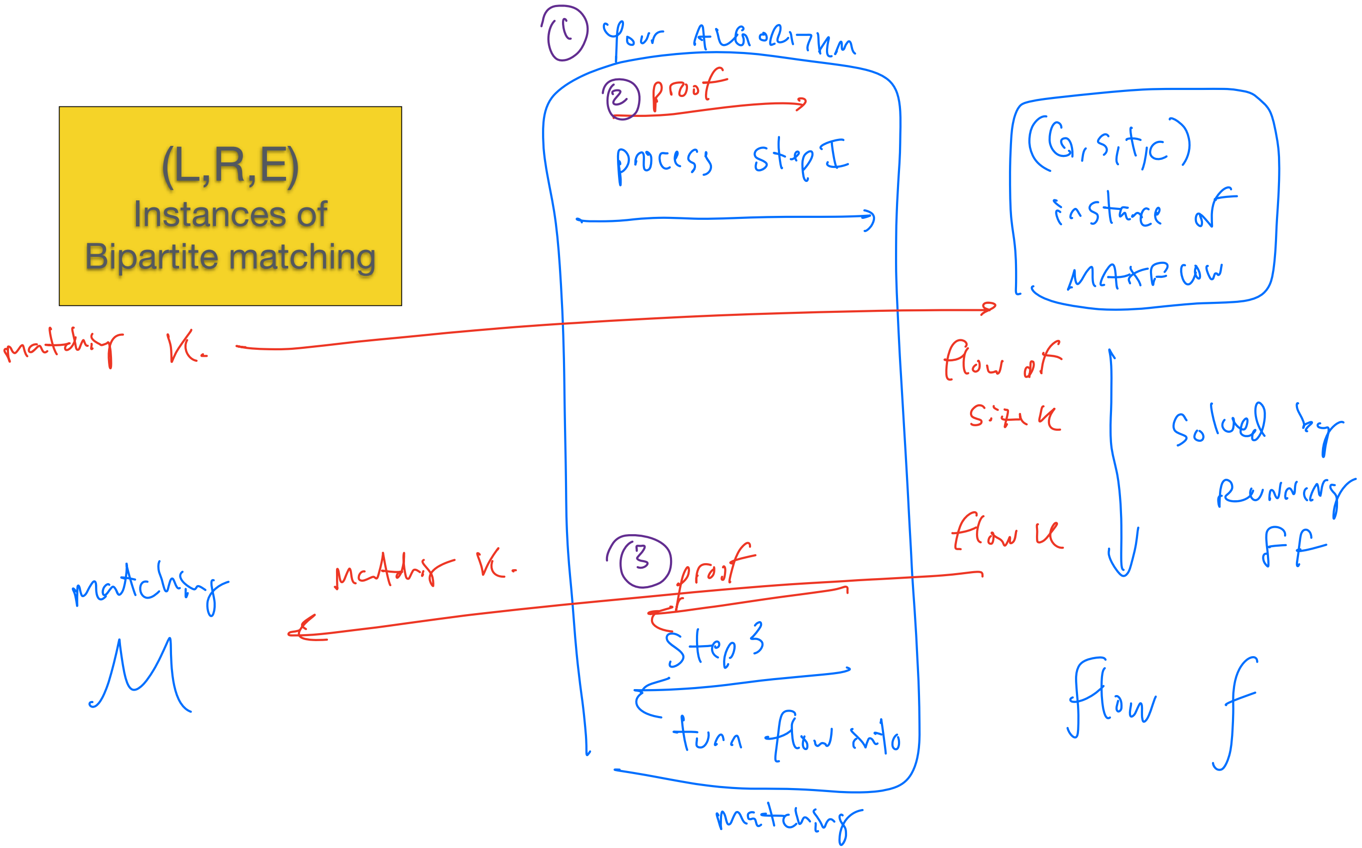
Now define $M: \{ e \mid f(e)=1 \text{ and } e=(x,y) \text{ s.t } x \in L\ y \in R \}$

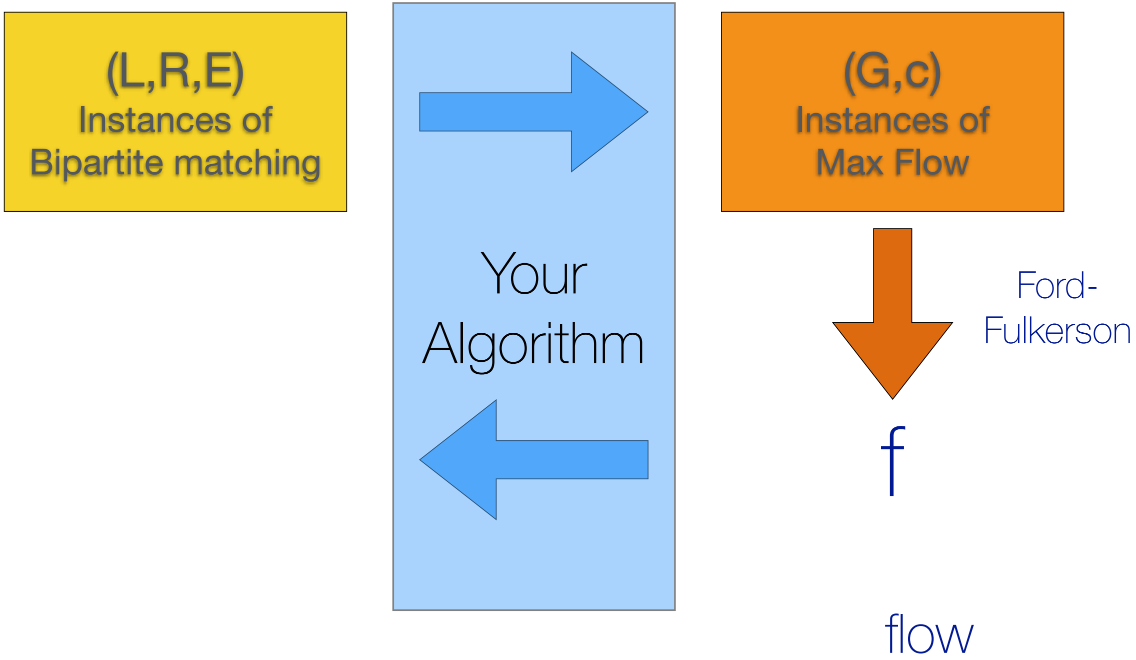
Prove that M is a matching.

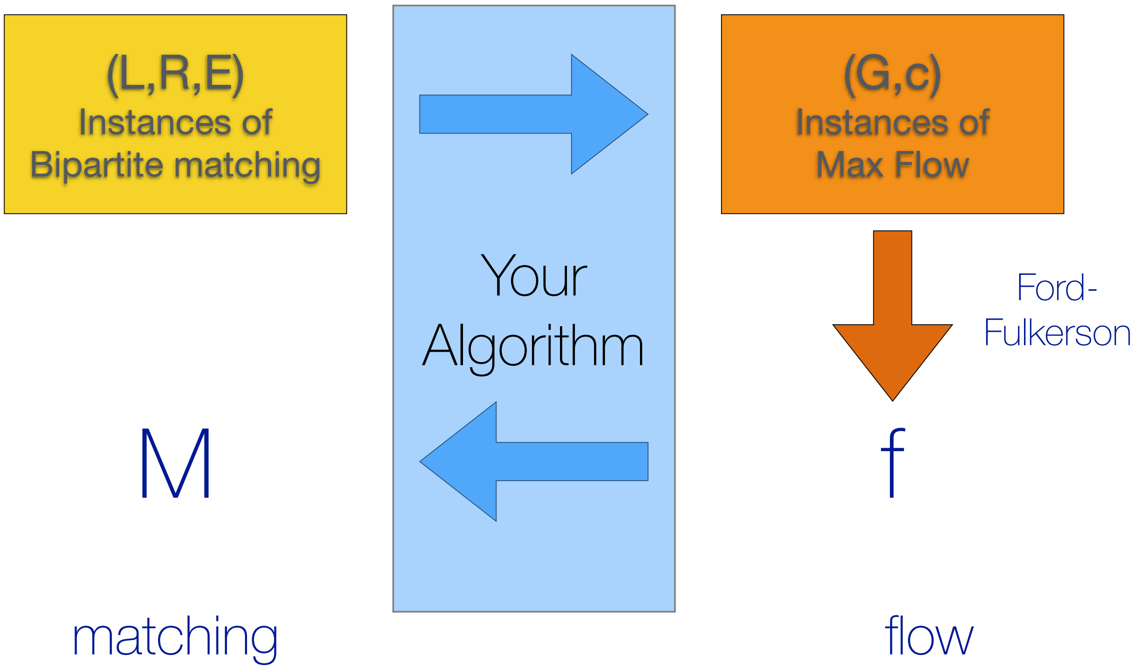
- All flows are integral & capacity $c(e)=1$. so $f(e) = 0$ or 1 for $e \in E$.

Thus for all $v \in L$, ($v \in R$) 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
matching M
matching K.
① Your ALGORITHM
② proof
process step I
(G,s,t,c)
instance of
MAXFLOW
flow of
size K
solved by
Running
FF
flow K
flow f
③ proof
matching K.
Step 3
turn flow into
matching

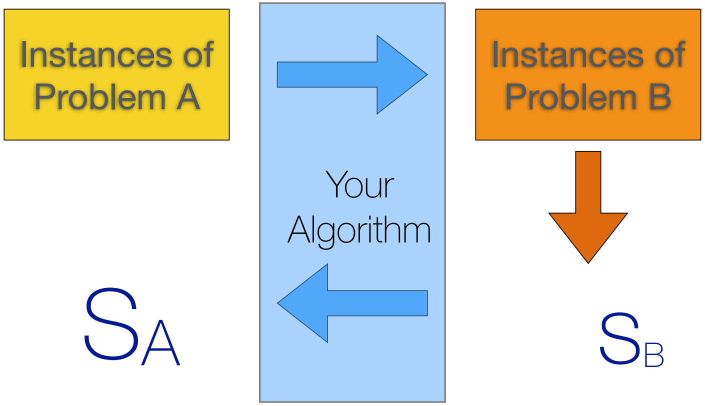

(L,R,E)
Instances of
Bipartite matching
Your
Algorithm
(G,c)
Instances of
Max Flow
Ford-
Fulkerson
f
flow


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

Reduction

A reduces in time f(n) to problem B.

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



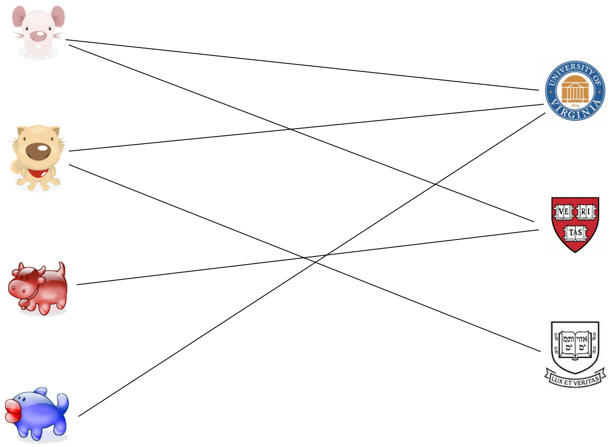
Runs in time f(n)

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

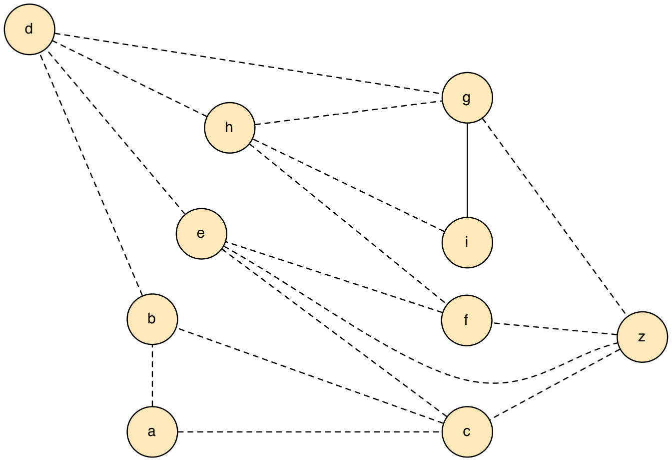
$$\exists c, d$$

$$T(\text{PROBLEM}_a(n)) \le f(n) + cT(\text{PROBLEM}_b(dn))$$

Maximum bipartite



edge-disjoint paths

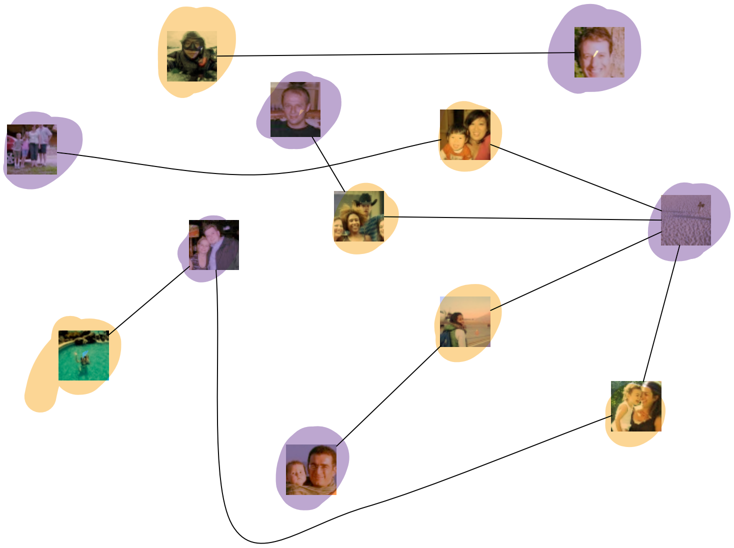


$\textsc{max}\text{BIPARTITE} <_{E+V} \textsc{max}\text{FLOW}$

$\textsc{max}\text{EDGEDISJ} <_{E+V} \textsc{max}\text{FLOW}$

↳ "reduces in time E+V to MAXflow"

party problem



Graph of friends who do not get along with one another

an edge corresponds to a conflict b/w your friends.

- goal: find the largest set of your friends that get along w/ea other

Def: independent set

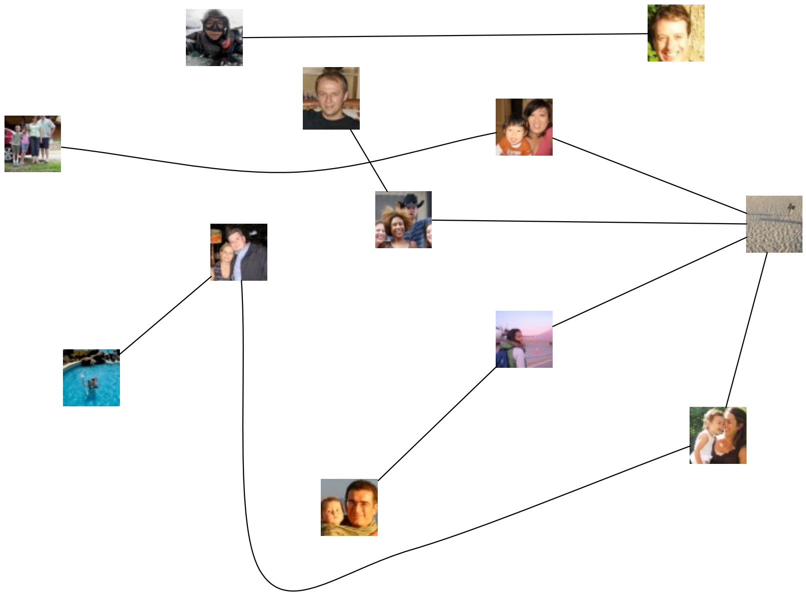
goal: find the largest independent set in $G=(V,E)$

INDSET: A subset of the vertices $S \subseteq V$ such that no two nodes $x,y \in S$ have an edge between them, i.e. $(x,y) \notin E$.

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:

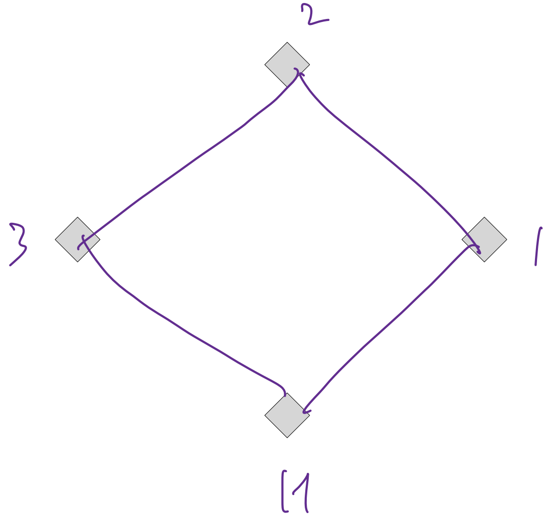
Given a graph $G=(V,E)$,

goal:

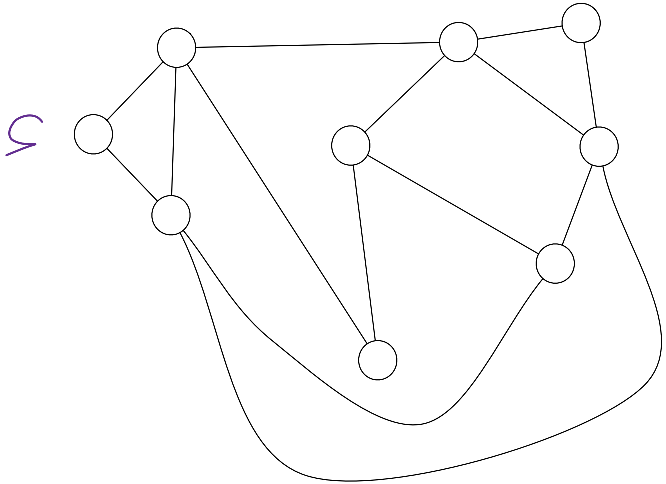
Given a graph $G=(V,E)$, find the largest or max independent set.

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

subset of nodes $C \subseteq V$ such that

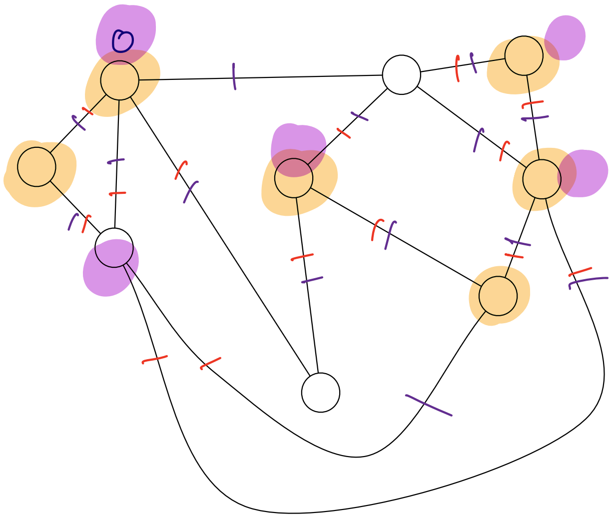
for each $e = (x,y) \in E$, either

$x \in C$ or $y \in C$.

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



6

5

goal:

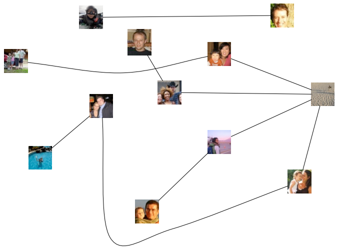
given a graph G, $= (V, E)$ find the

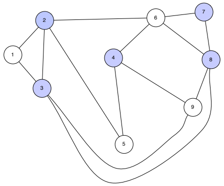
smallest set cover.

goal:

given a graph G,

Find the minimum sized vertex cover for G.





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

A solution to VC can be used to solve INDSET.

What is required to show this reduction?

① your algorithm

IND set instance

$G=(V,E)$

② proof

instance of

min vertex

cover

any solution

③ proof

independent

set cover

C

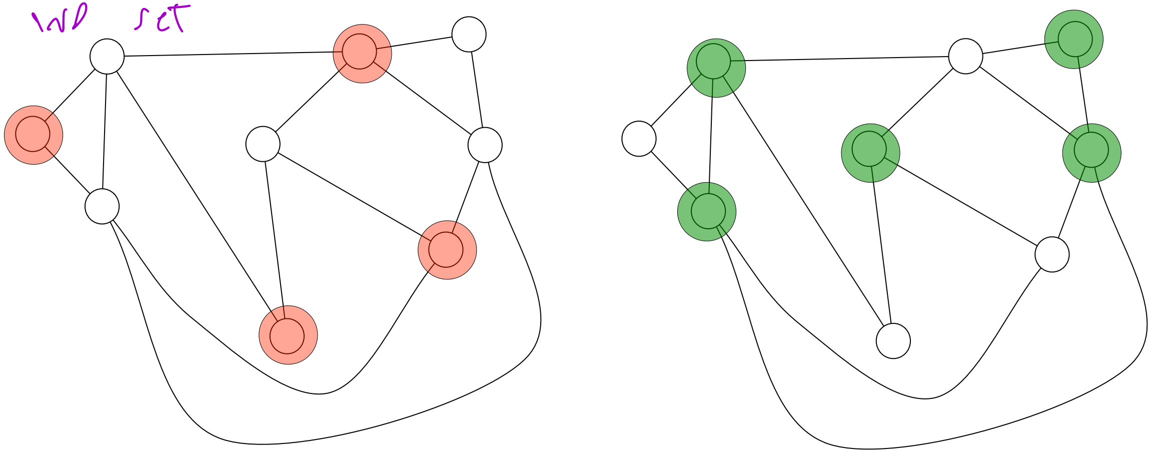
$O(V)$

Thm: Set S is a vertex cover for $G = (V, E)$ if and only if $(V - S)$ is an ind set of G.

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.

ind set



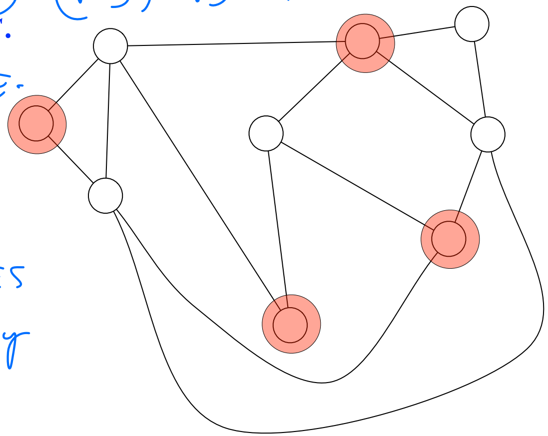
Thm: set S is an independent set of G iff V-S is a vertex cover.

S indset $\Rightarrow$ $(V-S)$ is a vertex cover.

goal is to show $(V-S)$ is a vertex cover.

Suppose S is an independent set.

Consider any edge $e=(x,y) \in E$.



① If $x \in S$, then $y \notin S$

$\Rightarrow$ $y \in (V-S)$. edge e is covered by $(V-S)$

② If $x \notin S$, then $x \in (V-S)$

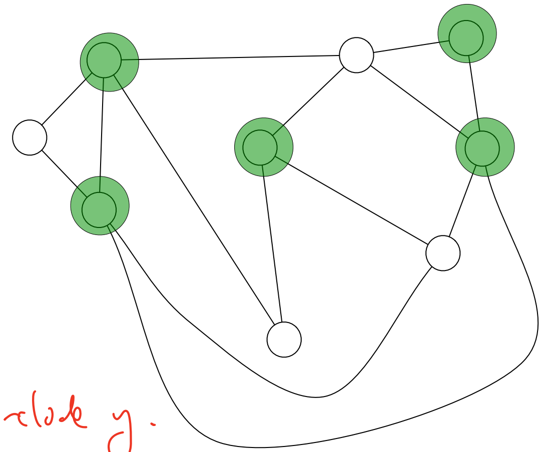
and again, e is covered.

This holds for <u>every</u> edge $e \in E$. So $(V-S)$ is a vertex cover.

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

SUPPOSE V-S IS A VC.

Consider any $x \in S$ and any edge $e = (x, y) \in E$.



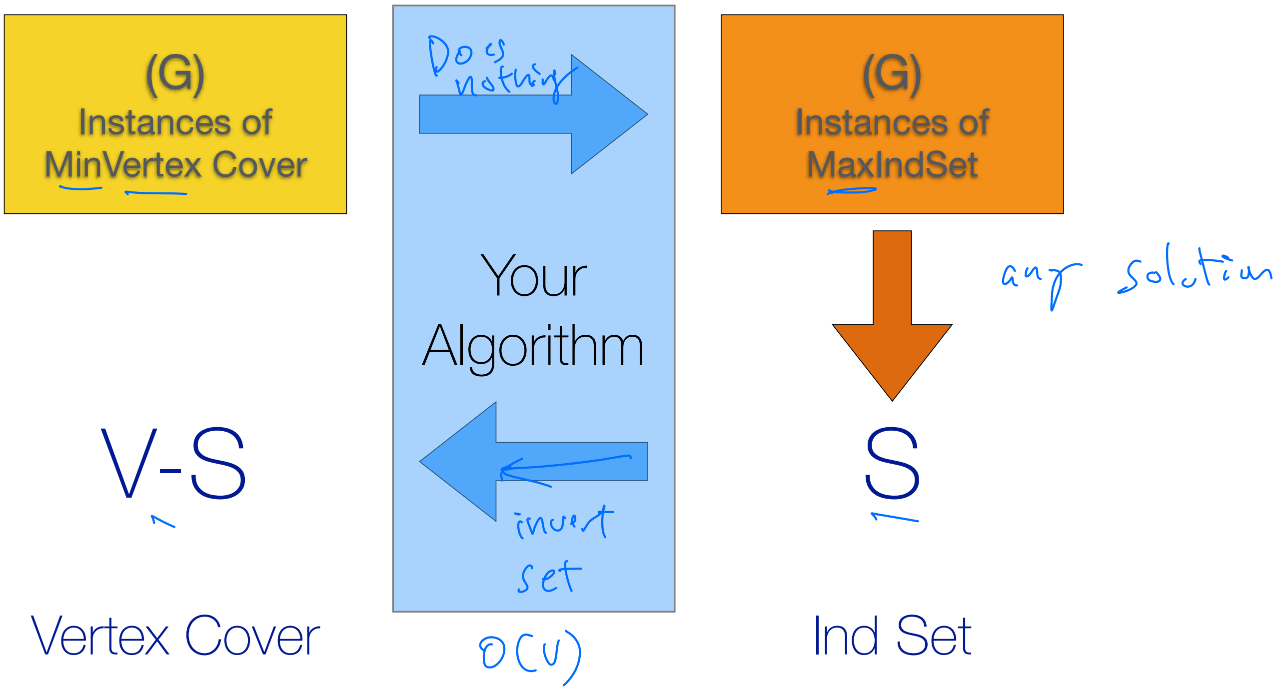
show that S is an IND set.

① Since $x \notin (V-S)$, then $y \in (V-S)$

why? Because $(V-S)$ is a vertex cover, so it must include y.

② This implies that $y \notin S$.

This is true for every $x \in S$ and every incident edge $e = (x, y) \in E$. $\Rightarrow$ S is independent set.


(G)
Instances of
MinVertex Cover
Does nothing
Your
Algorithm
invert
set
(G)
Instances of
MaxIndSet
any solution
V-S
S
Vertex Cover
O(V)
Ind Set

3sat problem

input: Boolean formula in 3CNF form: conjunction of clauses, each clause includes 3 variables

example: (a or b or c) AND (d or $\bar{a}$ or $\bar{b}$) AND (...)

output: find an assignment to all the variables that makes the formula TRUE.

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.

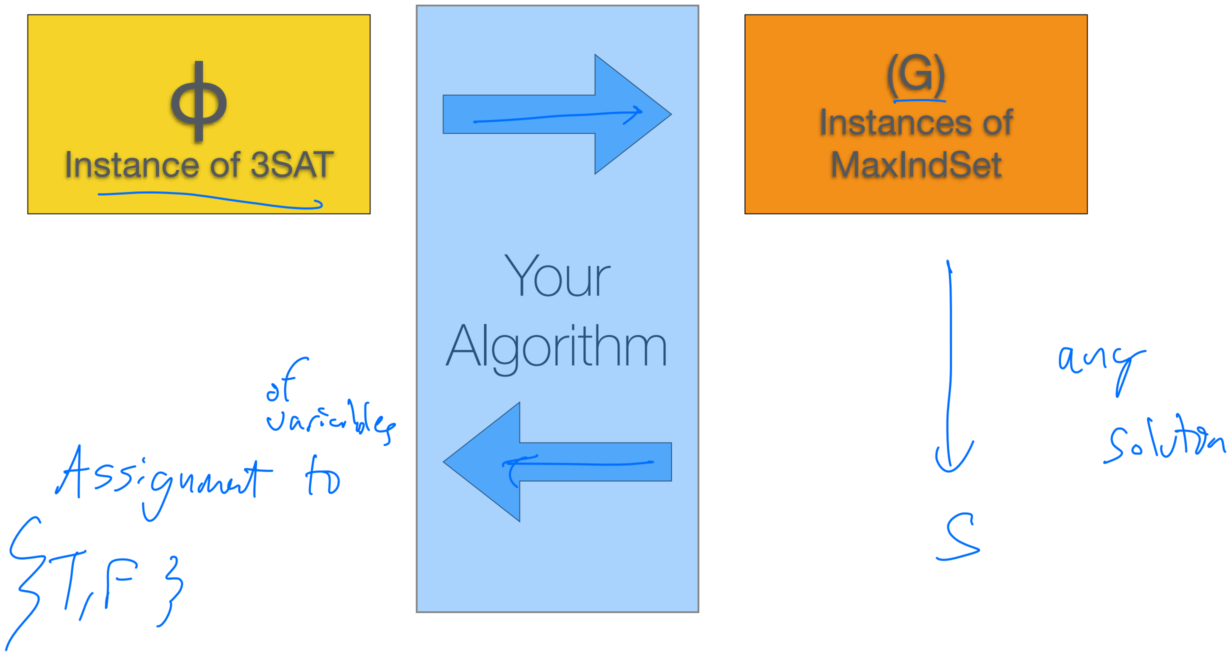
3sat example

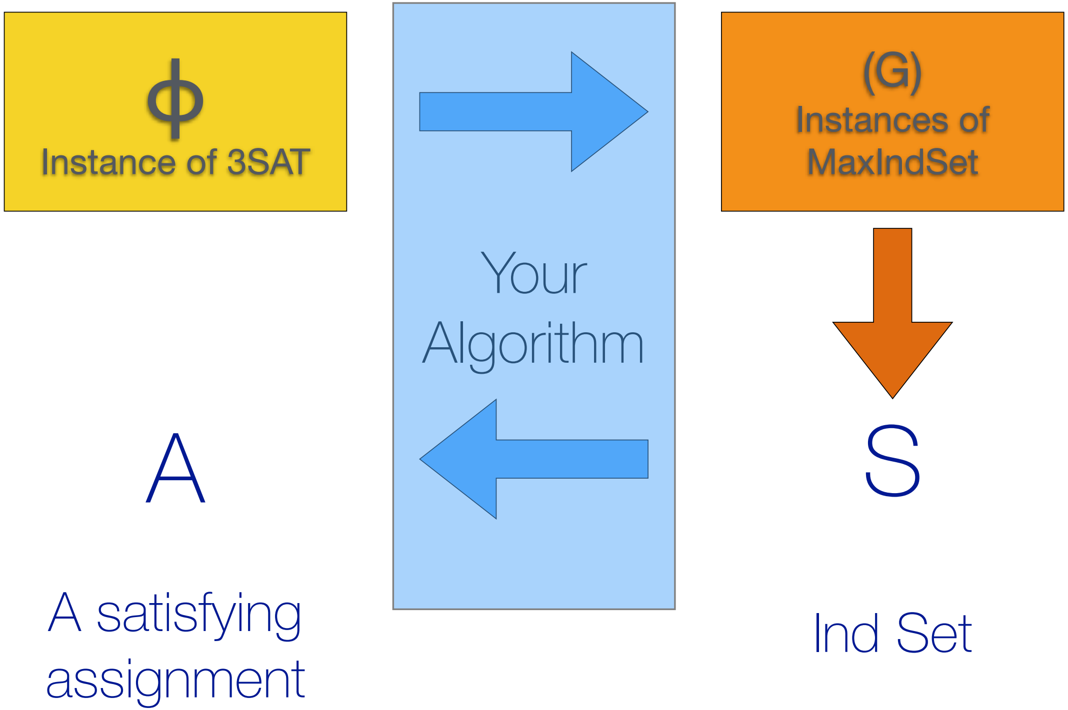
$$(x \vee y \vee z) \wedge (x \vee \overline{y} \vee y) \wedge (u \vee y \vee \overline{z}) \wedge (z \vee \overline{x} \vee u) \wedge (\overline{x} \vee \overline{y} \vee \overline{z})$$

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

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

what must we do to?


ϕ
Instance of 3SAT
Your Algorithm
(G)
Instances of MaxIndSet
of variables
Assignment to
{T,F}
any solution
S


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

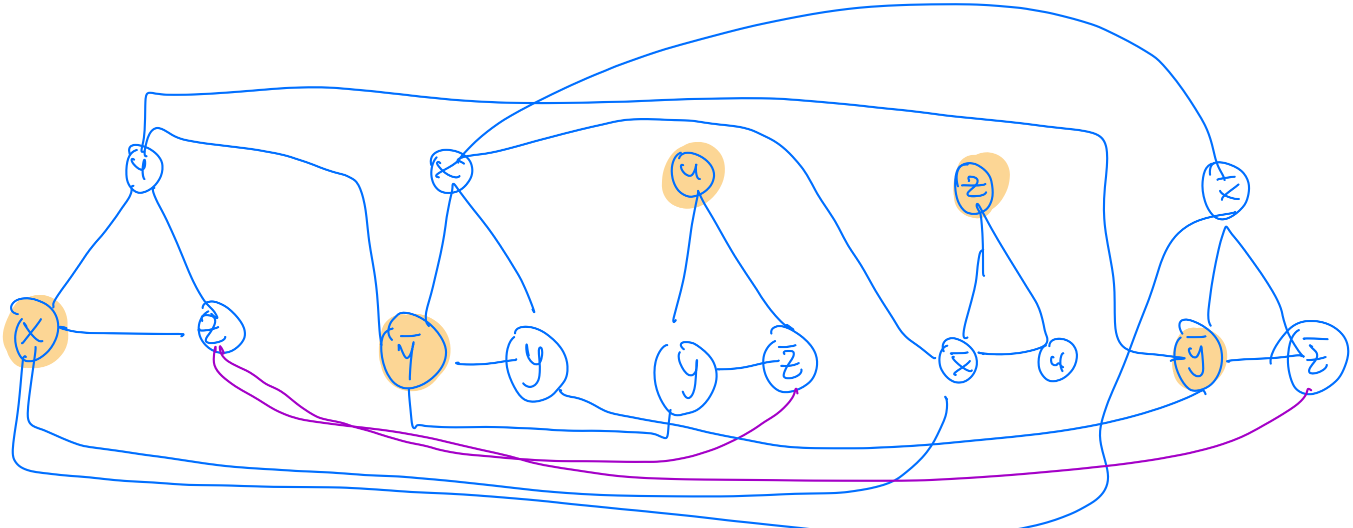
These arguments often follow a common pattern.

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

3SAT $\leq_p$ INDSET

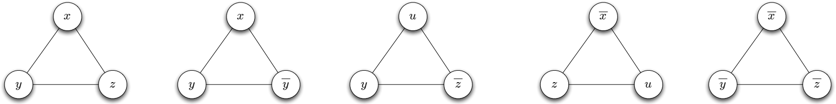
K clauses

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

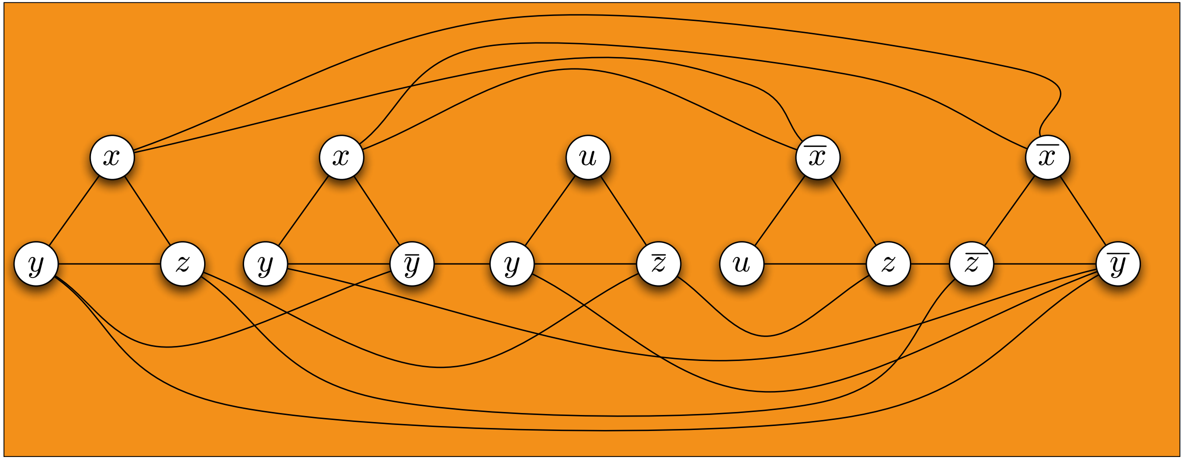


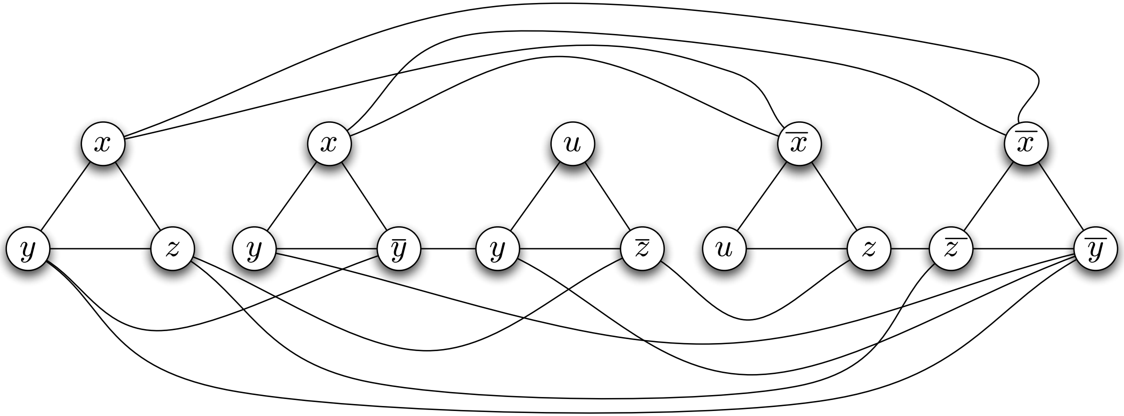
3SAT $\leq_p$ INDSET

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



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



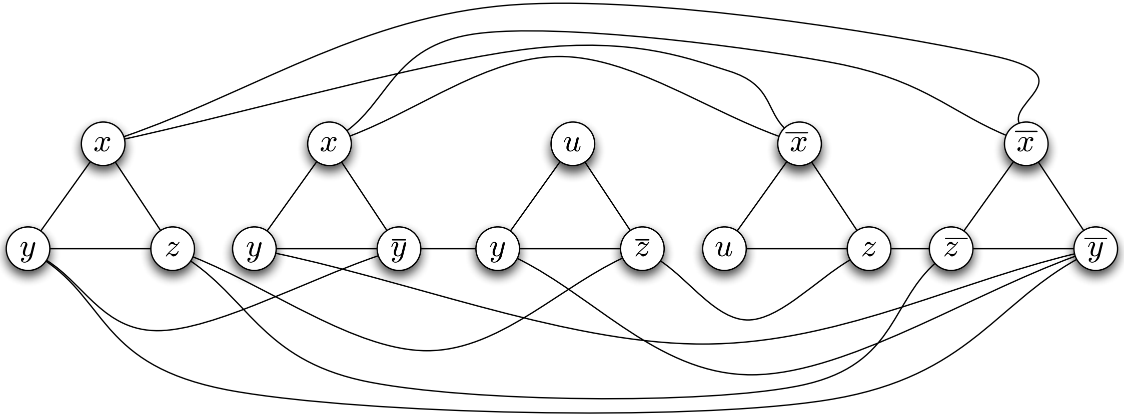


$\forall$ clause

$\phi \in \text{SAT} \Longrightarrow \exists$ an independent set G of size K

① each clause has some variable which makes it true. select 1 such variable for ea clause, and add its corresponding node to S.

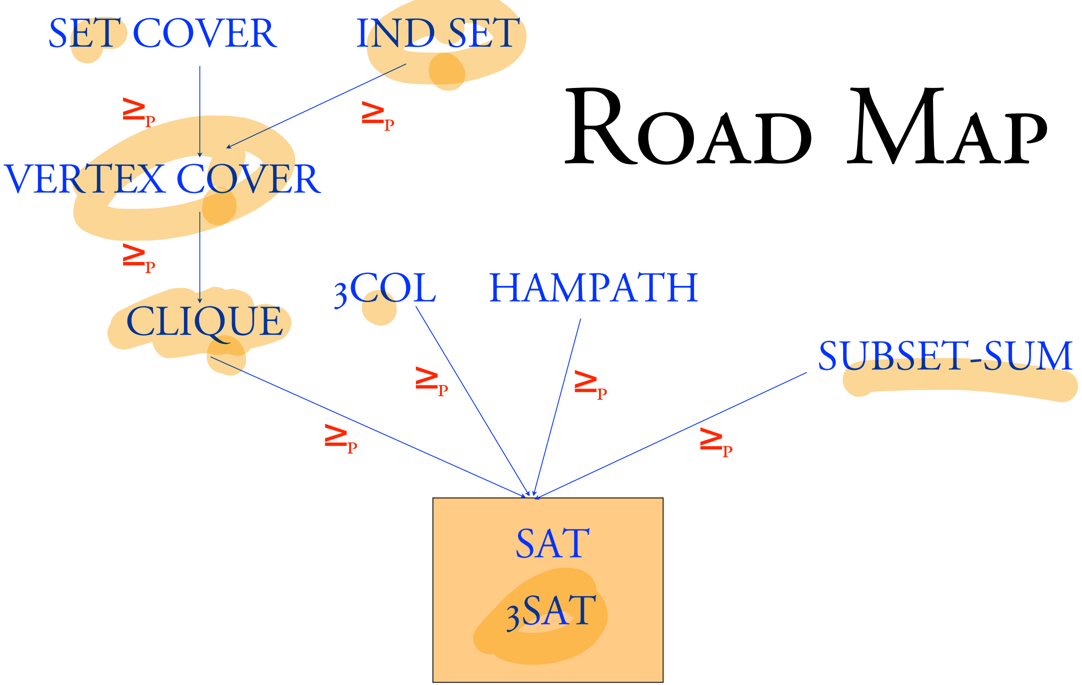
② S is an independent set. If $x \in S$, x was true, $\overline{x}$ was false and thus no instance of $\overline{x}$ could be chosen to be added to S. $\Rightarrow$ each triangle has exactly 1 selected node.



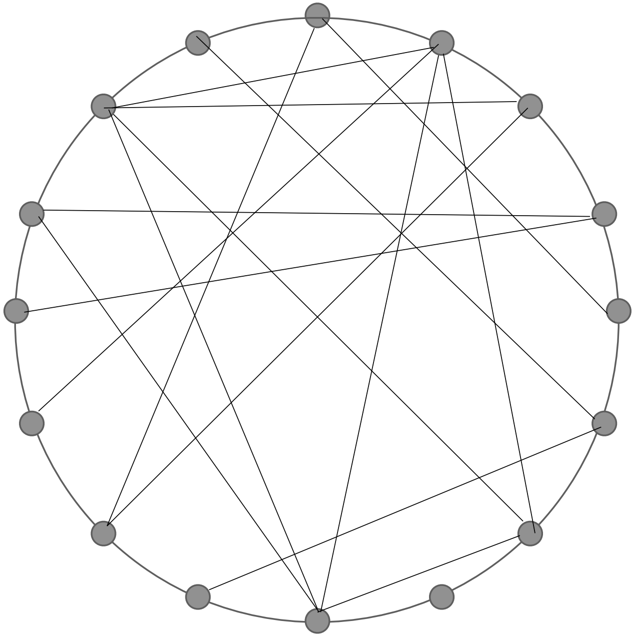
$(G, k) \in$ INDSET $\Longrightarrow$ (for you at home) L20

the original formula is satisfiable.

Road Map

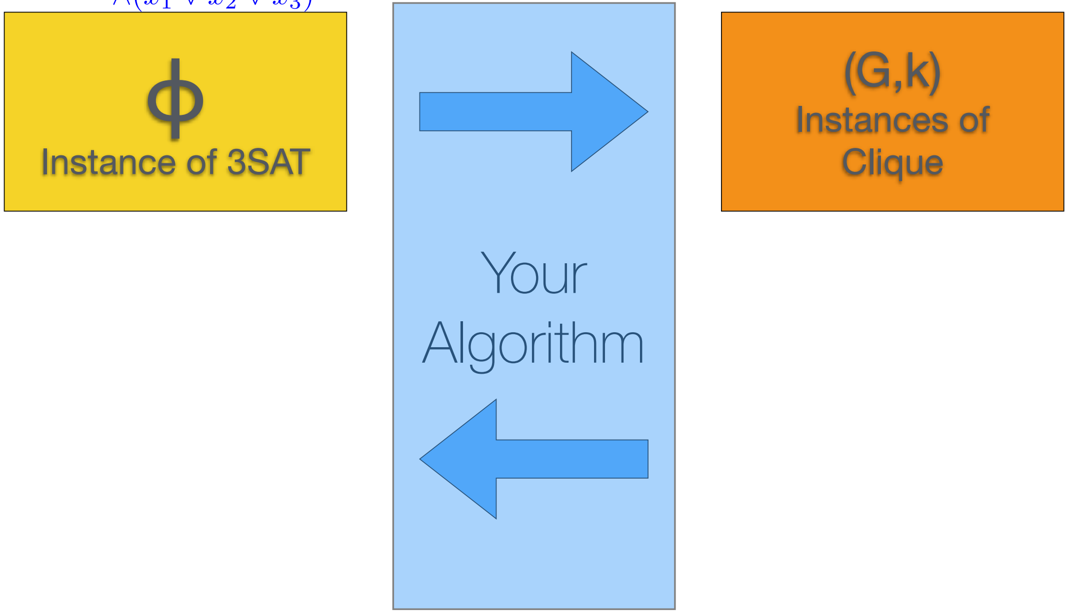

SET COVER
IND SET
≥P
≥P
VERTEX COVER
≥P
CLIQUE
3COL
HAMPATH
SUBSET-SUM
≥P
≥P
≥P
≥P
SAT
3SAT

clique

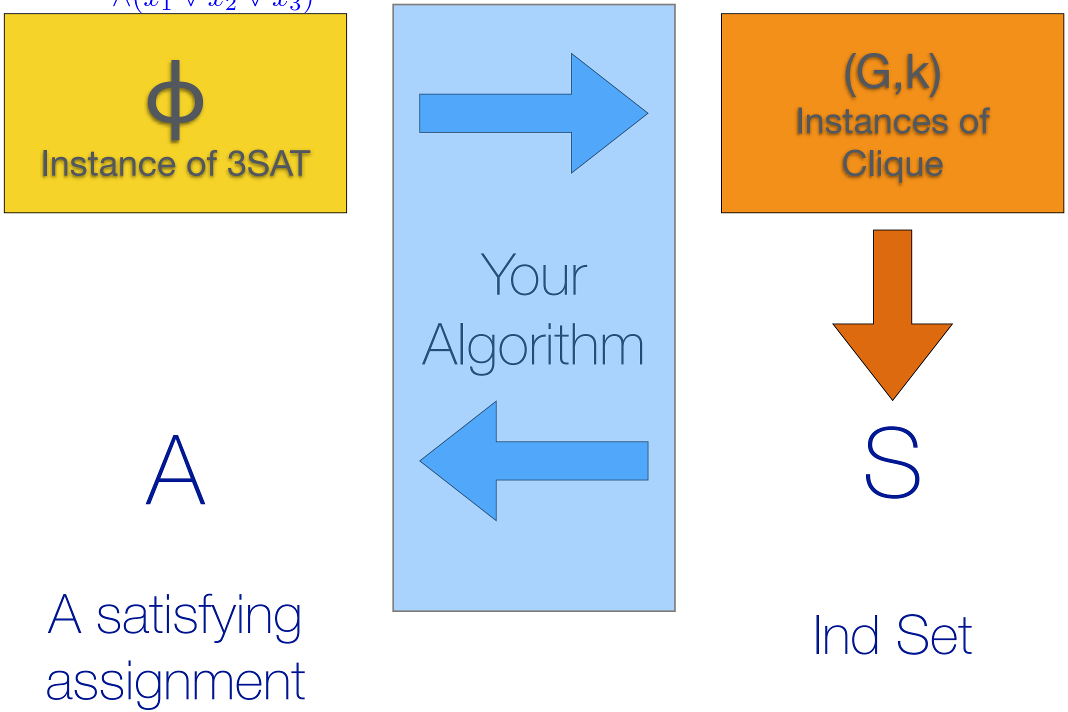


clique $= \{$

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



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



CLIQUE

FORMULA

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

$\xrightarrow{\text{F}}$

GRAPH, K

K = # CLAUSES

CLIQUE

FORMULA

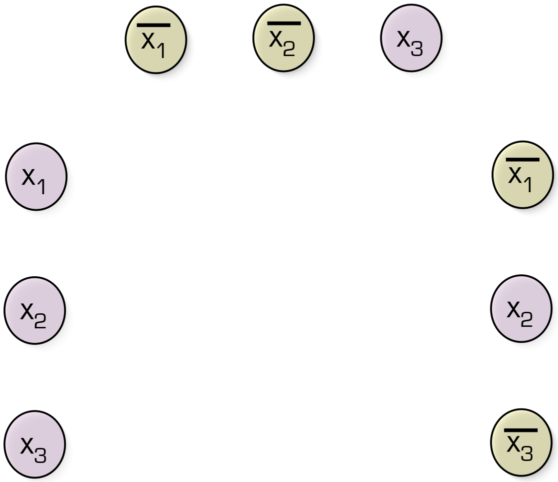
$$\phi = \begin{array}{l}(x_1 \vee x_2 \vee x_3) \\ \wedge(\overline{x_1} \vee \overline{x_2} \vee x_3) \\ \wedge(\overline{x_1} \vee x_2 \vee \overline{x_3})\end{array}$$

F

GRAPH, K

K = # CLAUSES

CREATE 3 NODES/CLAUSE



CLIQUE

FORMULA

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

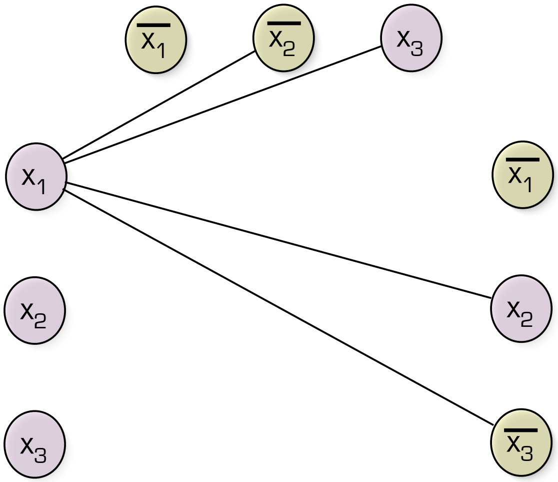
F

GRAPH, K

K = # CLAUSES

Create 3 nodes/clause

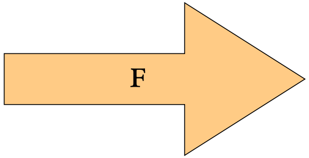
Connect nodes to
“non-opposites”



CLIQUE

FORMULA

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

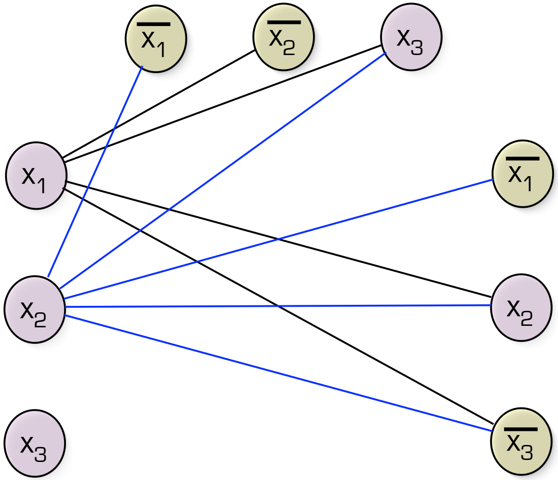


GRAPH, K

K = # CLAUSES

Create 3 nodes/clause

Connect nodes to
“non-opposites”



CLIQUE

FORMULA

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

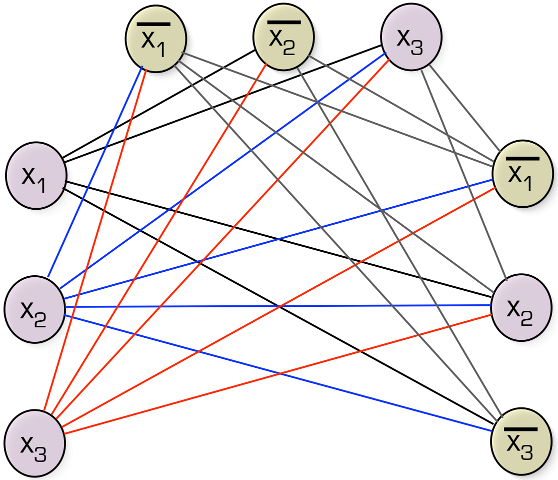
F

GRAPH, K

K = # CLAUSES

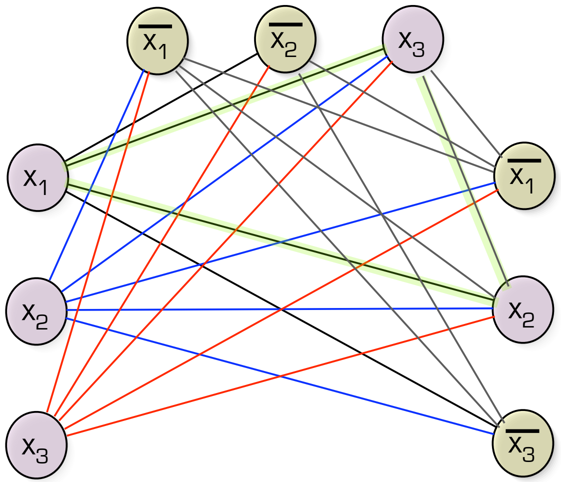
Create 3 nodes/clause

Connect nodes to
"non-opposites"



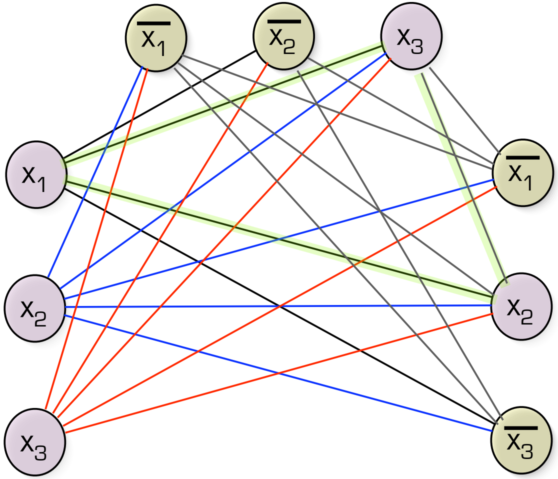
CLIQUE

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



CLIQUE

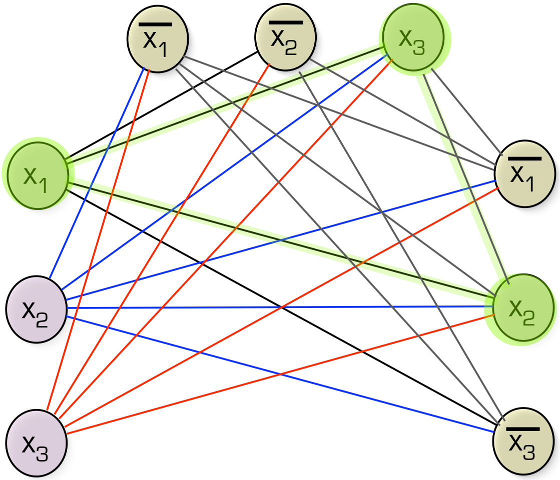
$$\phi = \begin{array}{l}(x_1 \vee x_2 \vee x_3)\\ \wedge(\overline{x_1} \vee \overline{x_2} \vee x_3)\\ \wedge(\overline{x_1} \vee x_2 \vee \overline{x_3})\end{array}$$



Satisfying assignment = 1 var/clause

CLIQUE

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

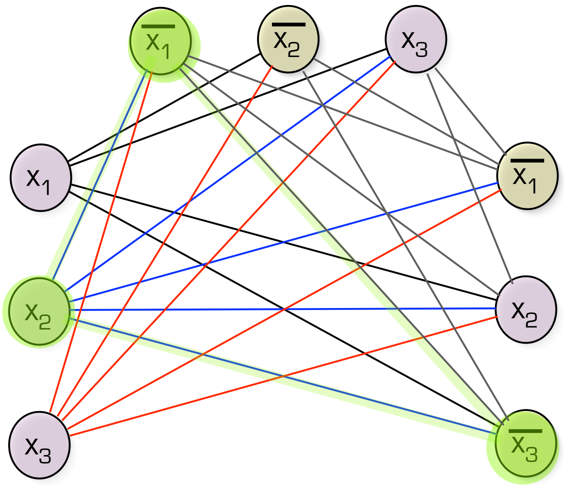


Satisfying assignment = 1 var/clause

k "non-opposite" connected nodes

CLIQUE

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

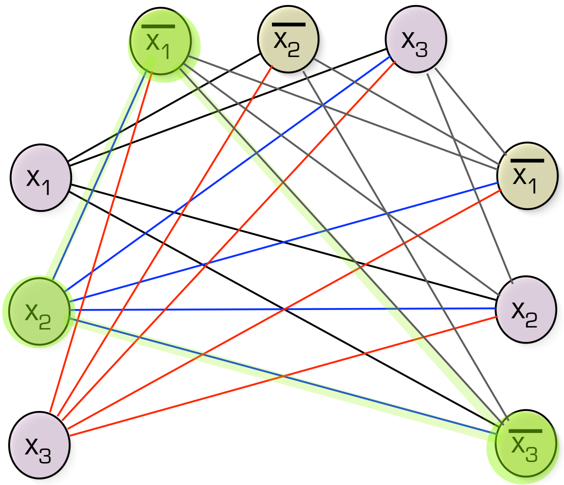


K-CLIQUE

1 NODE/CLAUSE IS TRUE

CLIQUE

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



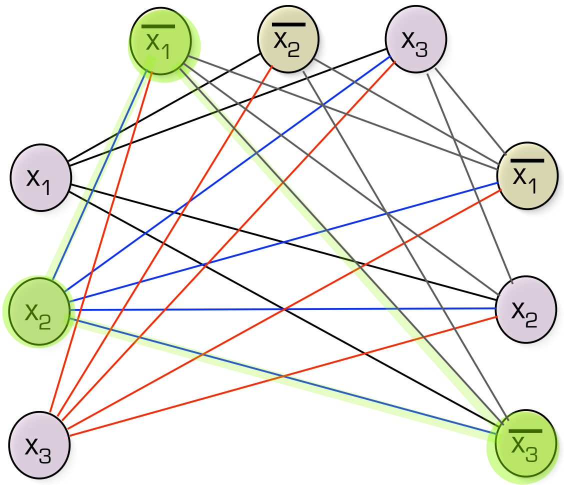
K-CLIQUE

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

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

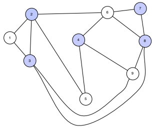
NP-Completeness

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

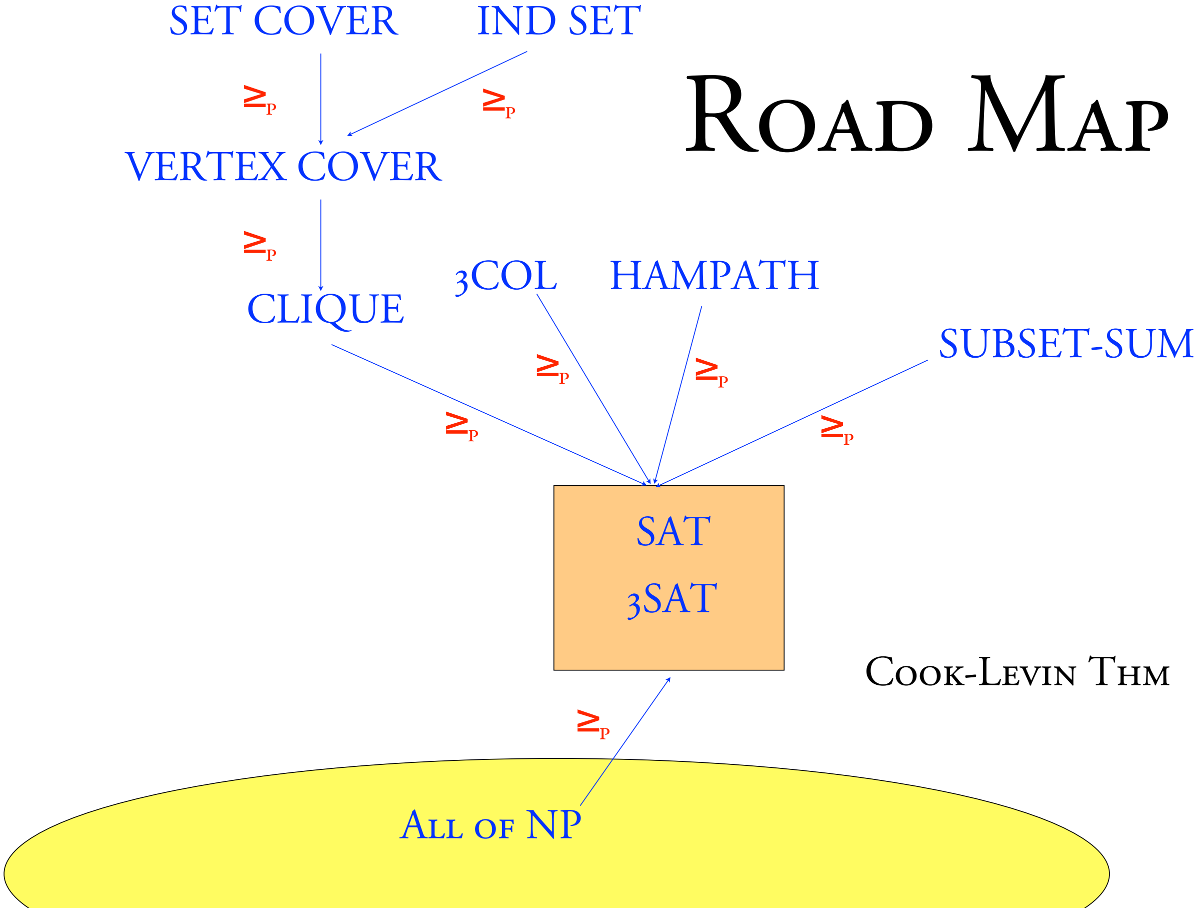
WHAT IS THE HARDEST PROBLEM IN NP?

Cook-Levin theorem

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

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

Road Map


SET COVER
IND SET
VERTEX COVER
CLIQUE
3COL
HAMPATH
SUBSET-SUM
SAT
3SAT
Cook-Levin Thm
All of NP
≥P
≥P
≥P
≥P
≥P
≥P
≥P
≥P