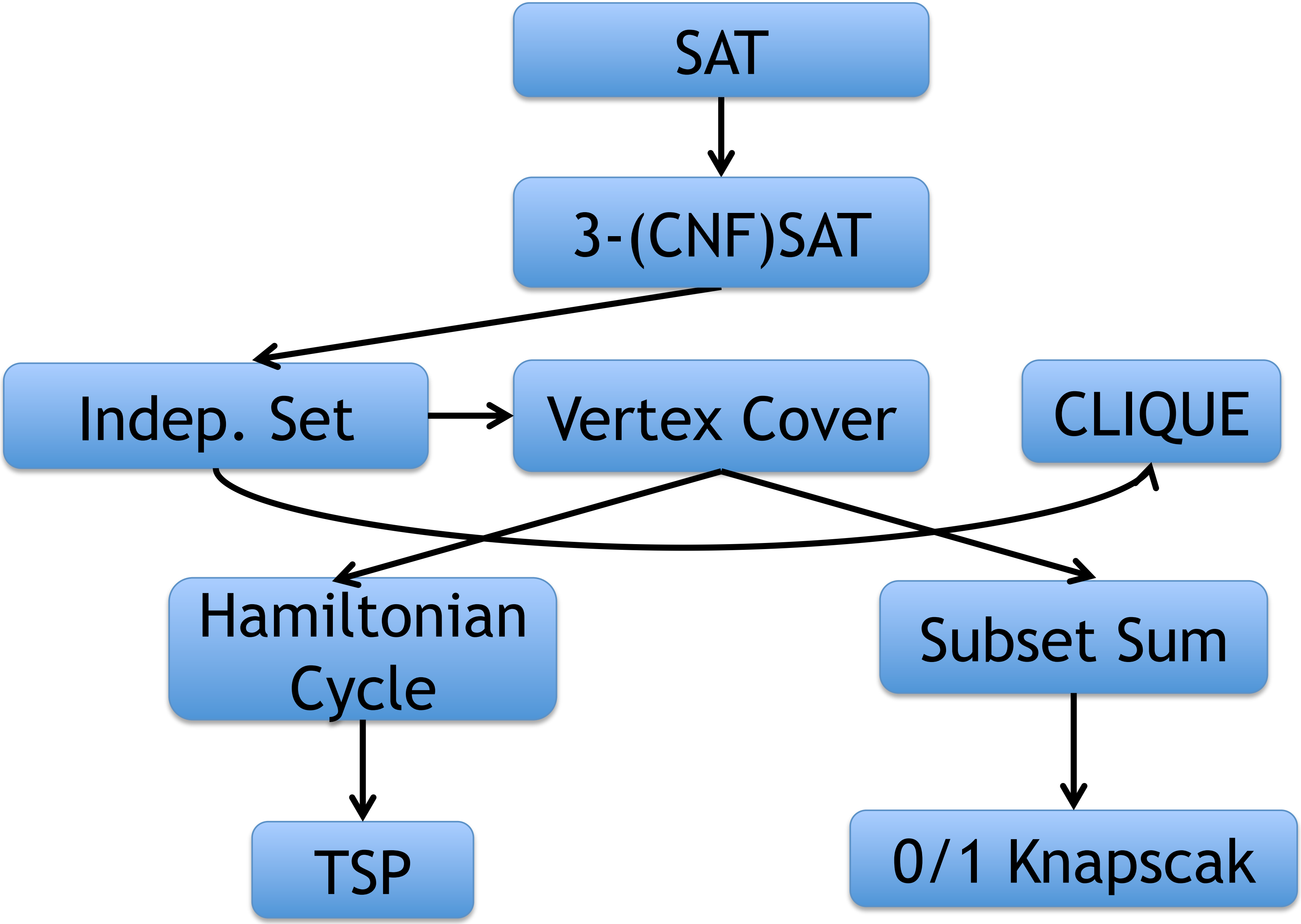
P, NP, NP-completeness 2

Reductions

Thu, March 28th, April 2nd

Overview of the next 2 Lectures



First: A bit of history on SAT

The First NP-Complete Problem

SAT: The First NP-Complete Problem

Input: A boolean formula φ consisting of:

n boolean variables $x_1, x_2, \ldots, x_n$

m boolean connectives: $\wedge$ (AND), $\vee$ (OR), $\neg$ (NOT), $\leftrightarrow$ (iff), $\rightarrow$ (implication), … (can be others)

and parantheses

Output: Is φ satisfiable?

I.e., are there true/false values to x_i that make φ true?

Example SAT Formulas

$$\varphi = (x_1 \rightarrow x_2) \land \neg x_2$$

Q: Is this satisfiable?

A: Yes

x_1	x_2	$(x_1 \rightarrow x_2) \land \neg x_2$
0	0	1
0	1	0
1	0	0
1	1	0

Example SAT Formulas

$$\varphi = (x_1 \rightarrow x_2) \land \neg x_2$$

Q: Is this satisfiable?

A: Yes

x_1	x_2	$(x_1 \rightarrow x_2) \land \neg x_2$
0	0	1
0	1	0
1	0	0
1	1	0

Example SAT Formulas

$$\varphi = (x_1 \rightarrow \neg x_2) \wedge \neg x_2$$

Q: Is this satisfiable?

A: Yes

x_1	x_2	$(x_1 \rightarrow \neg x_2) \wedge \neg x_2$
0	0	1
0	1	0
1	0	1
1	1	0

Example SAT Formula

$$\varphi = (x_1 \rightarrow \neg x_2) \wedge \neg x_2$$

Q: Is this satisfiable?

A: Yes

x_1	x_2	$(x_1 \rightarrow \neg x_2) \wedge \neg x_2$
0	**0**	**1**
0	1	0
1	0	0
1	1	0

Example SAT Formula

$$\varphi = ((x_1 \land x_2 \land x_3) \leftrightarrow (\neg x_1 \land x_3))$$

Q: Is this satisfiable?

A: No

x_1	x_2	x_3	φ
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	0

Recall 2 Criteria For NP-Completeness

1. C* has to be in NP.
2. Every other NP problem has to reduce to C*.

Criterion 1: Why is SAT in NP?

Can verify a solution $X^* = (x_1 = 0/1, \ldots., x_n = 0/1)$ is

linear time!

Just check if X^* makes φ true!

Criterion 2: Why does every NP problem reduce to SAT?

Method 1: Cook-Levin Theorem (1971)

(Won't show in class)

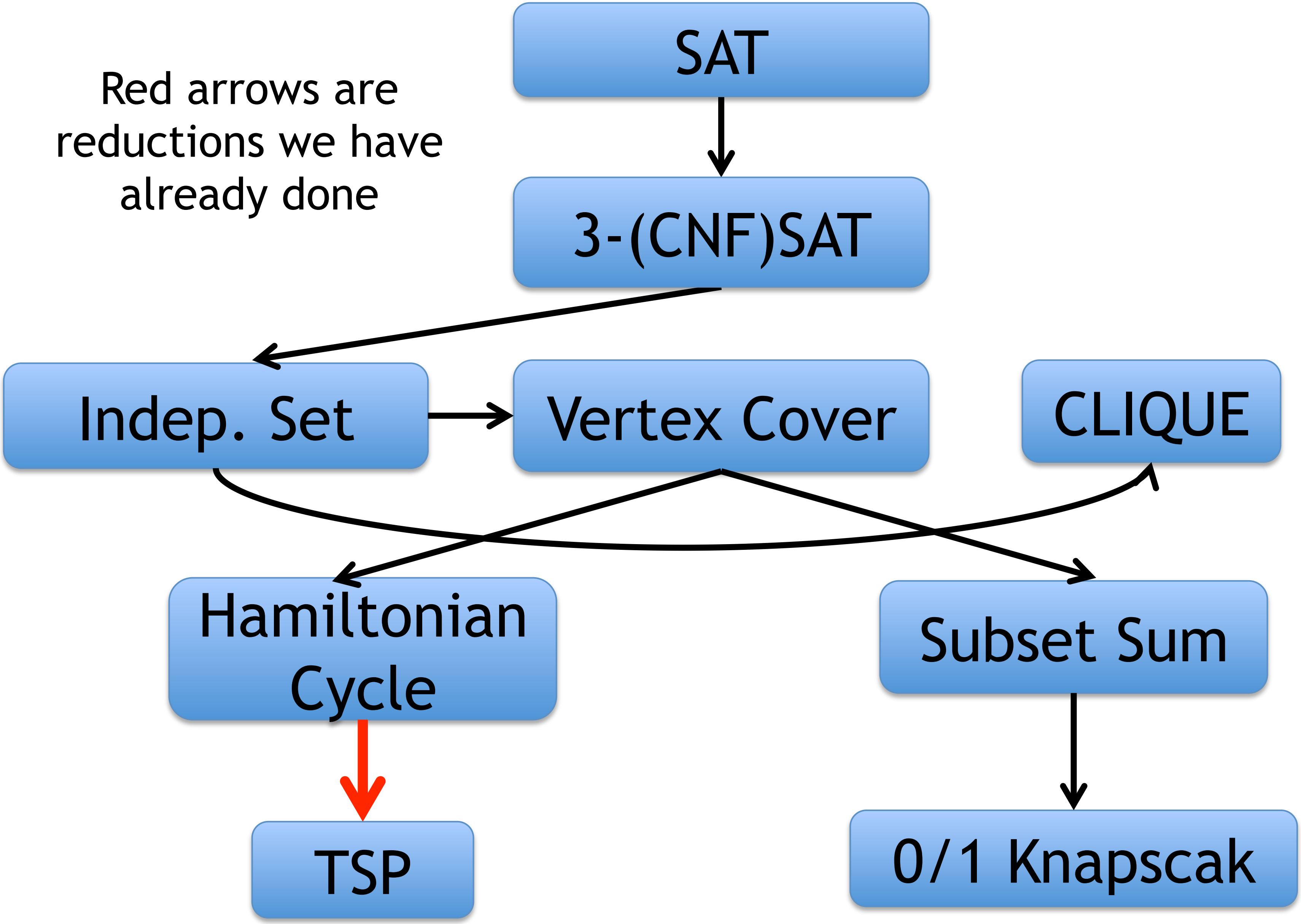
Method 2: Or you can show another known NP-Complete problem, e.g., CIRCUIT-SAT, reduces to SAT

(Also won't show in class)

Instead will show some reductions across another set of problems (all NP-Complete)

Reductions Tree

Red arrows are reductions we have already done

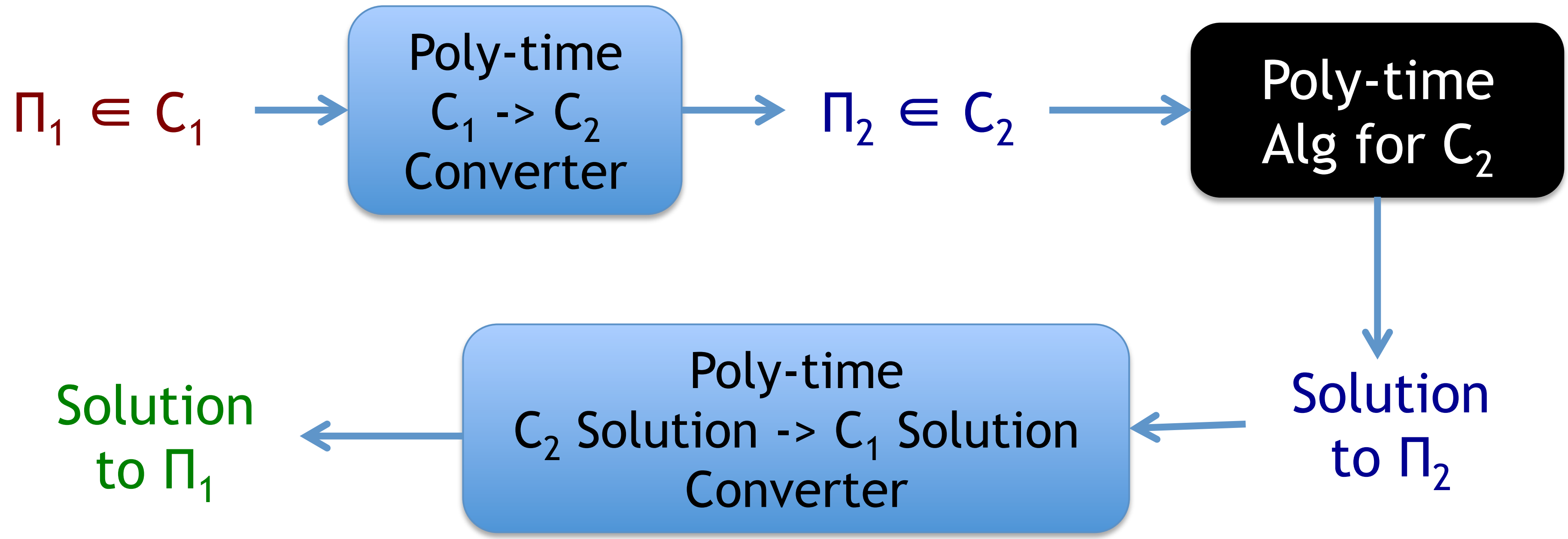


Recall Reductions: Showing C_2 is as hard as C_1

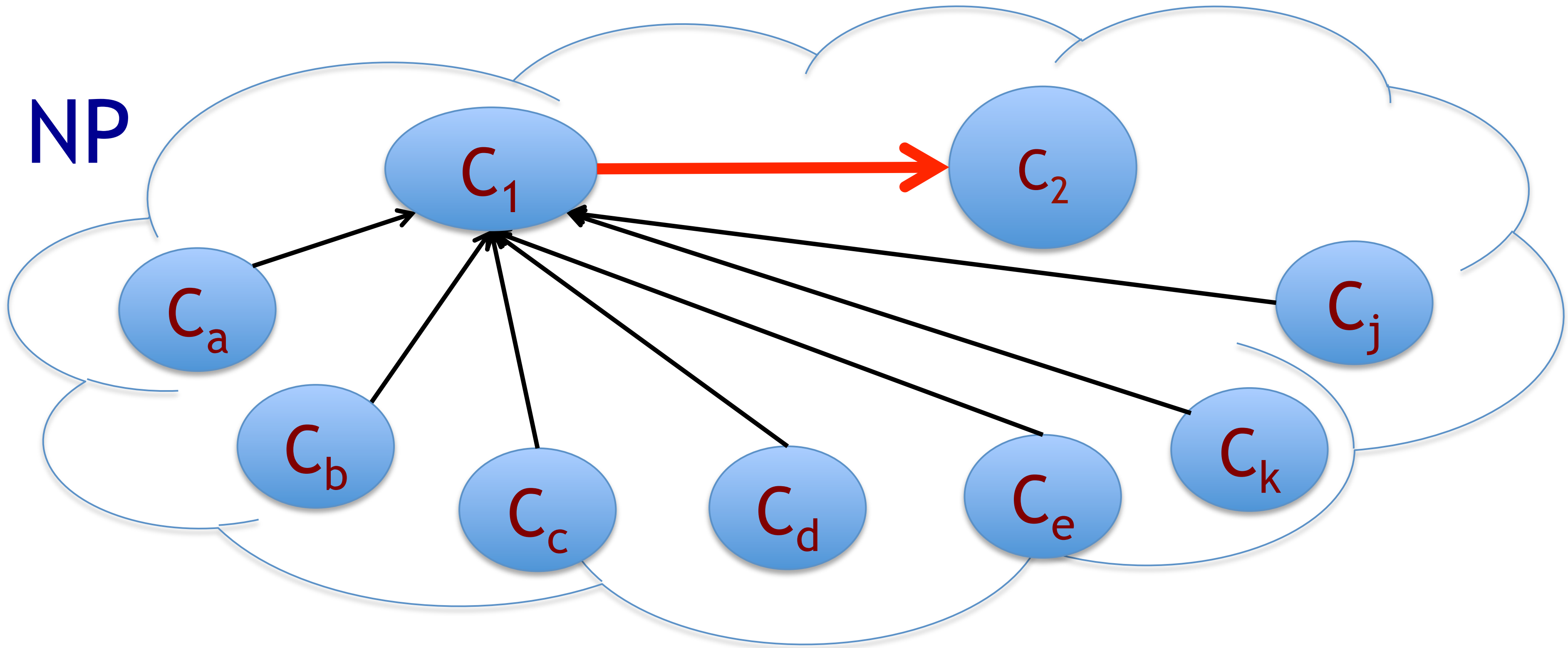
What does it mean for problem C_2 to be as hard as C_1?

Definition: C_1 *reduces* to C_2, ($C_1 \leq_p C_2$), if given a poly-time algorithm for C_2, we can solve C_1 in poly-time.

If C_1 reduces to C_2 => C_2 is "as hard as" C_1



An NP-complete C_1 reducing to C_2

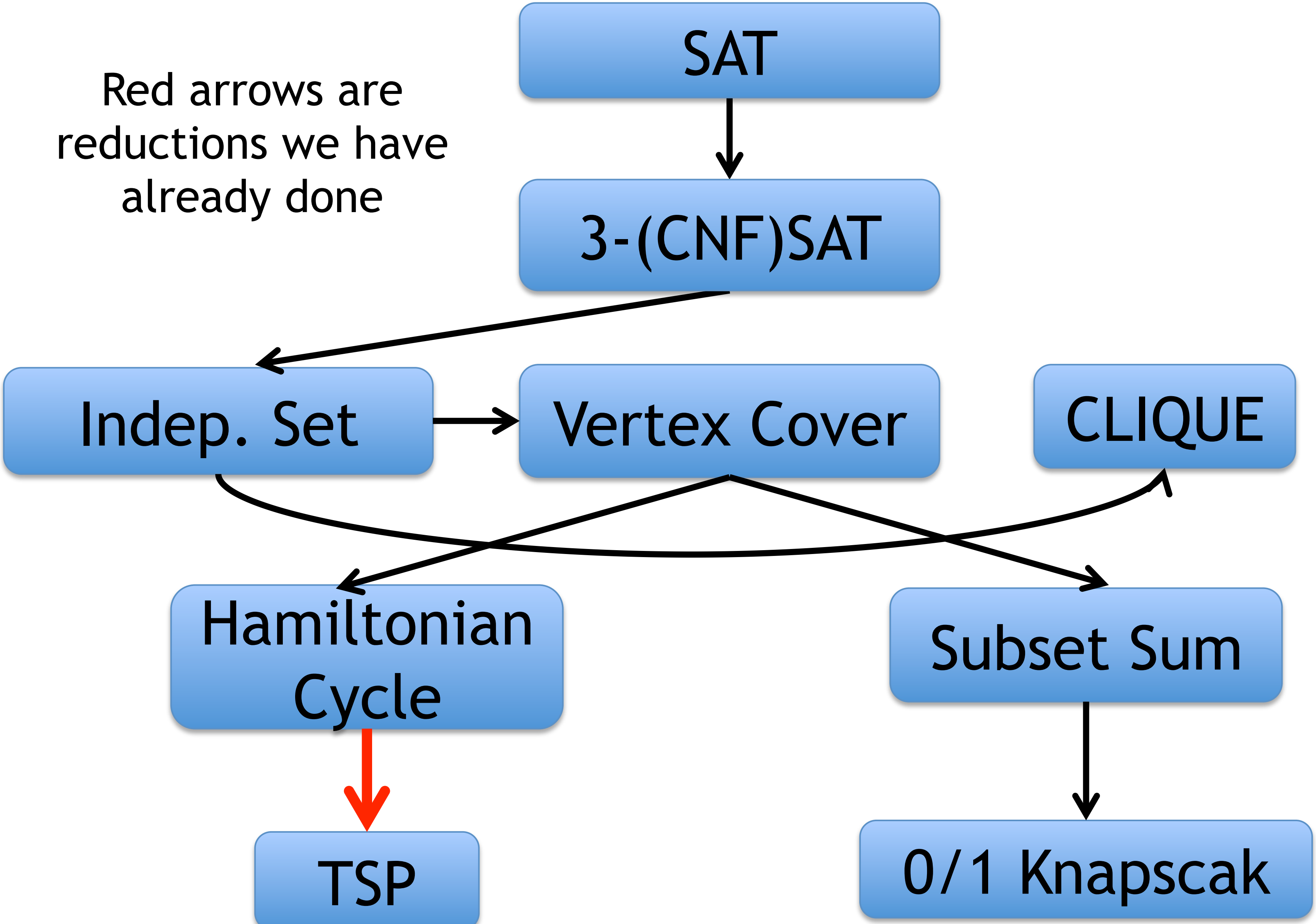


B/c reductions are transitive => All NP problems reduce to C_2

Therefore C_2 is also NP-complete!

Reductions Tree

Red arrows are reductions we have already done



Independent Set (IS)

Input: undirected graph G=(V, E) & an integer k

Output: "yes" iff $\exists$ subset $S \subseteq V$ of size $\geq k$ s.t.

no pair of vertices in S have an edge, i.e.,

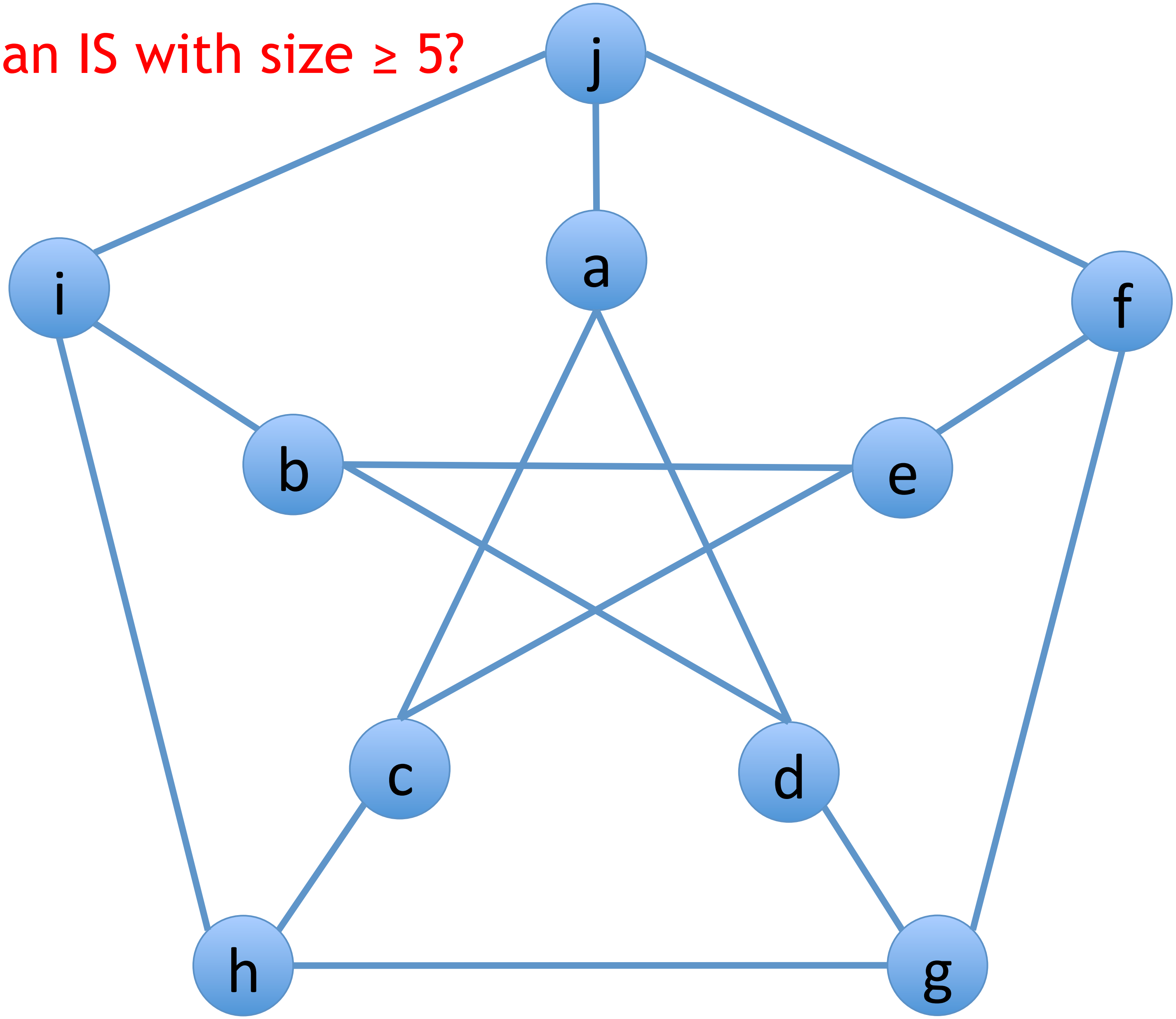
$\nexists\ (u, v) \in E$ s.t. $u \in S$ and $v \in S$

Recall: We solved this in linear time on line graphs!

IS Example

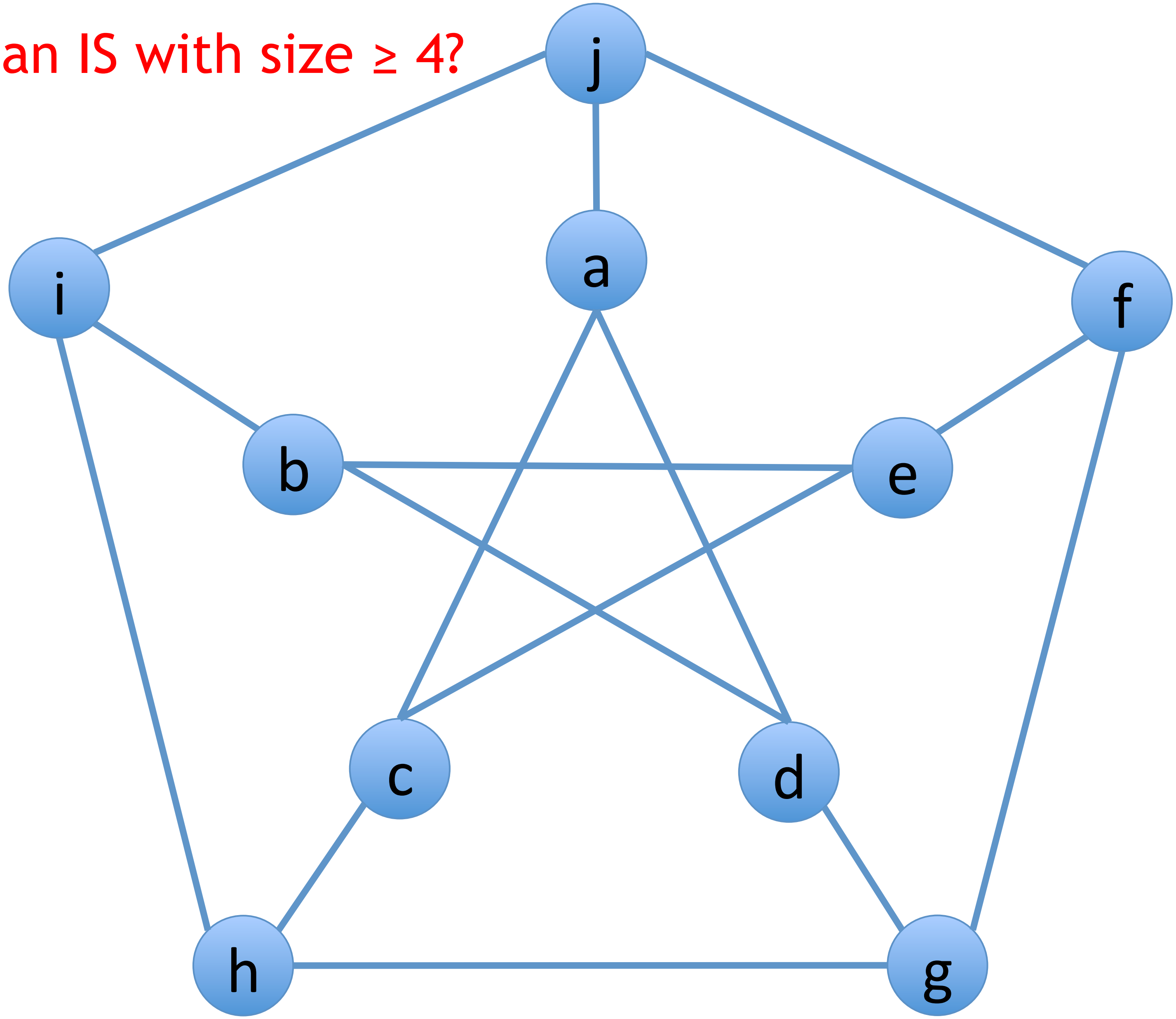
Q: ∃ an IS with size ≥ 5?

A: No



IS Example

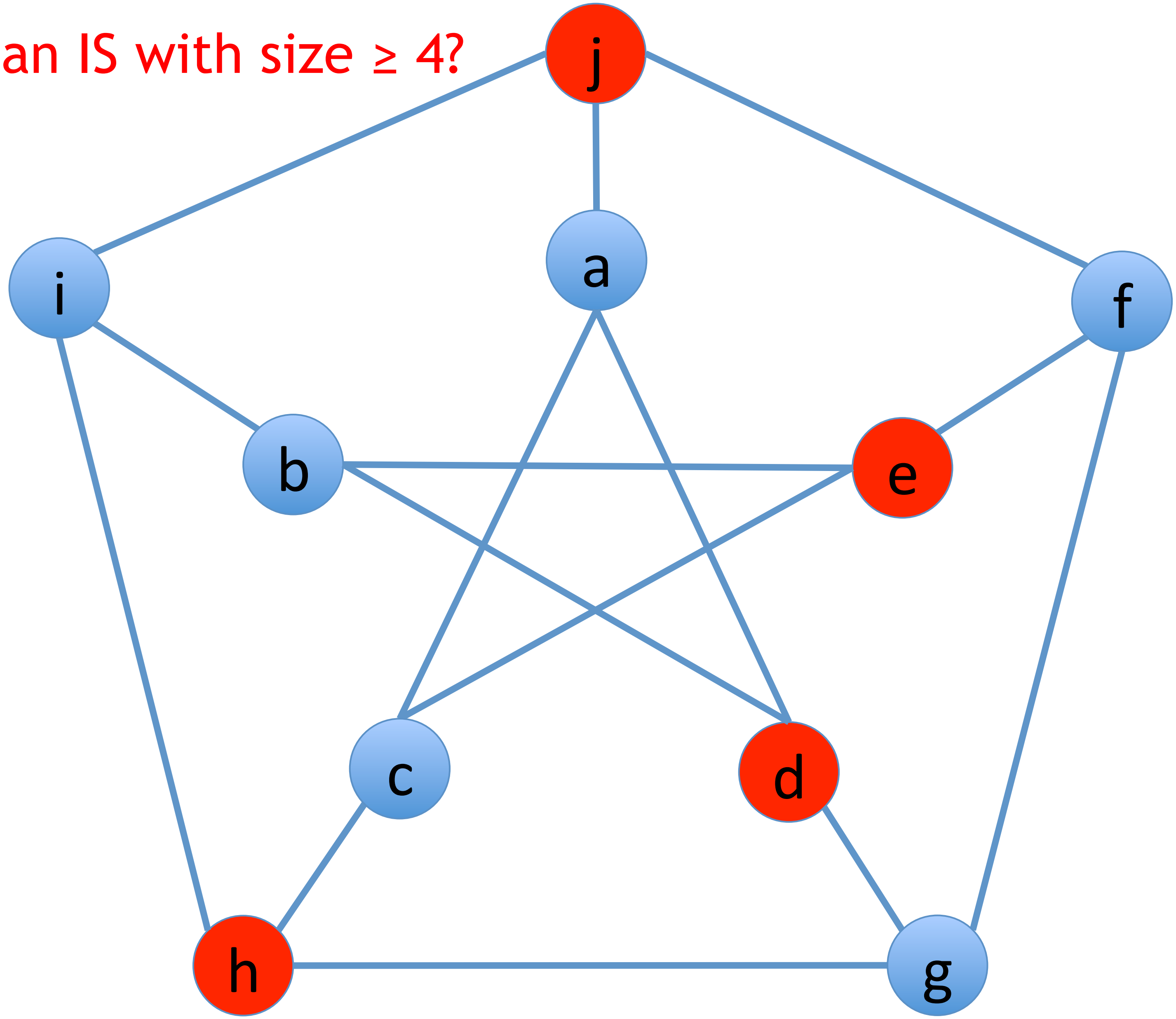
Q: ∃ an IS with size ≥ 4?



IS Example

Q: ∃ an IS with size ≥ 4?

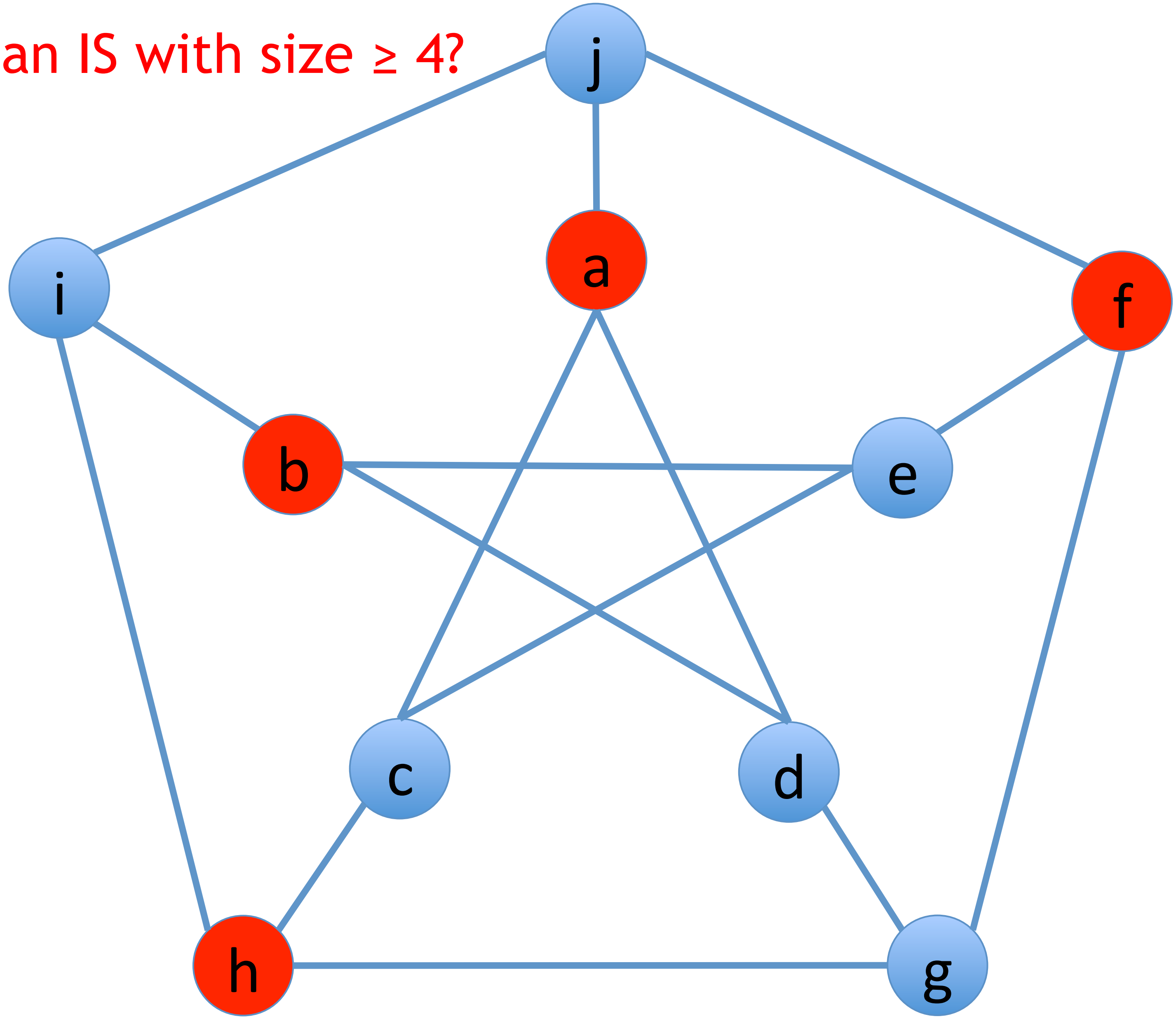
Yes



IS Example

Q: ∃ an IS with size ≥ 4?

Yes



Vertex Cover (VC)

Input: undirected graph $G=(V, E)$ & an integer k

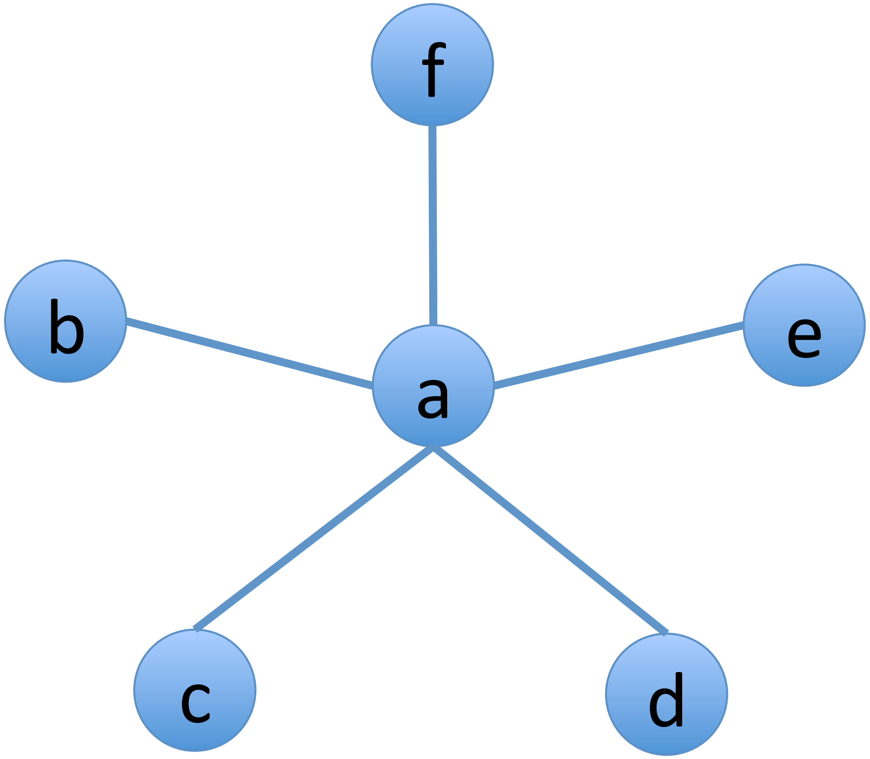
Output: "yes" iff $\exists$ subset $S \subseteq V$ of size $\leq k$ s.t.

$\forall (u, v) \in E$, either $u \in S$ or $v \in S$

(each edge is "covered" by at least one vertex $\in S$)

VC Example

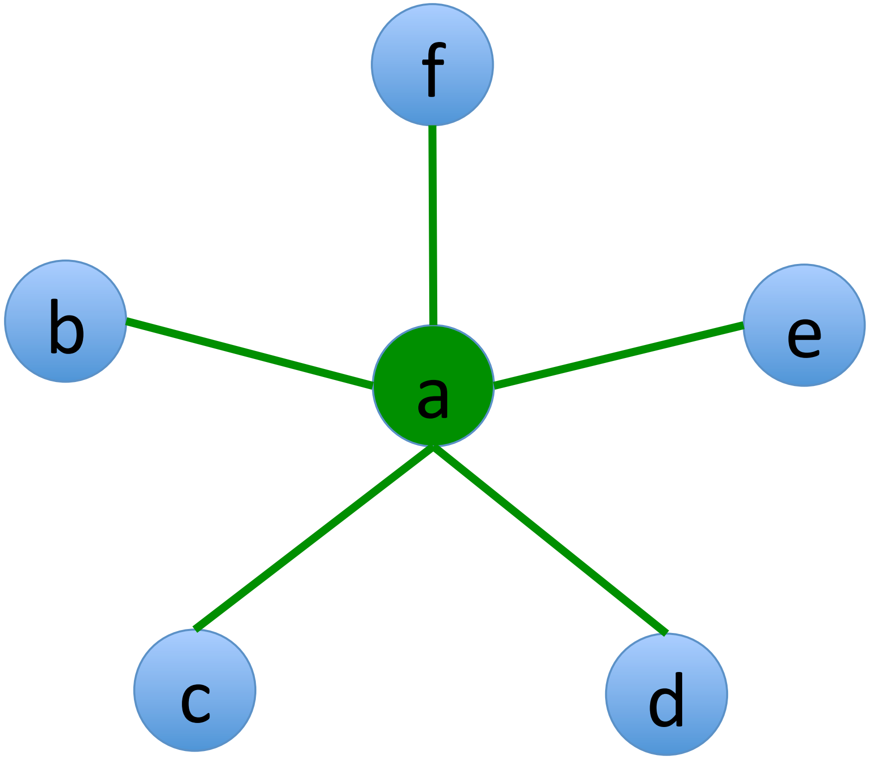
Q: ∃ a VS with size ≤ 1?



VC Example

Q: ∃ a VS with size ≤ 1?

A: Yes



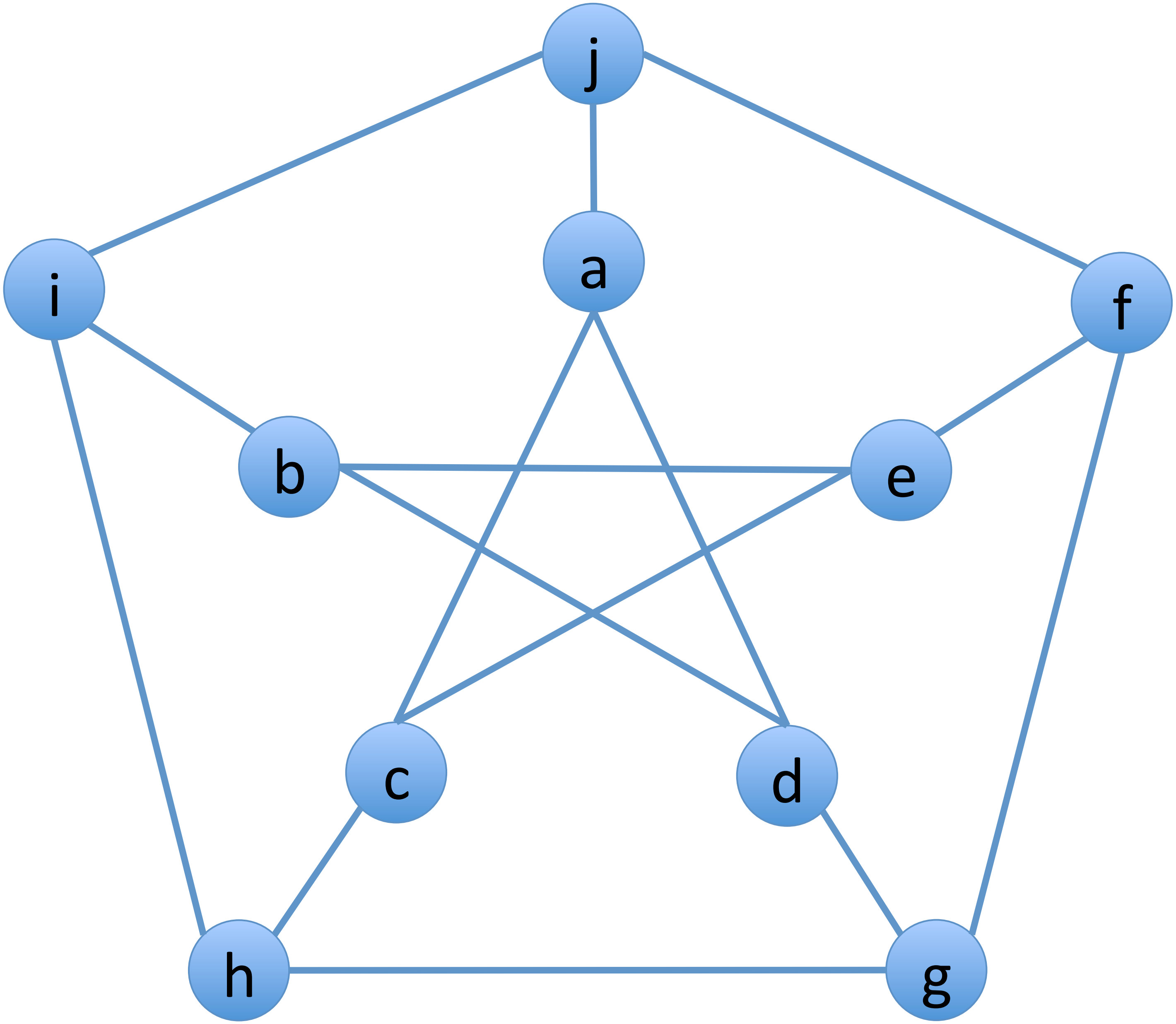
IS $\leq_p$ VC Proof Idea

$\exists$ an IS S with size = k

iff $\exists$ an VC with size = n-k

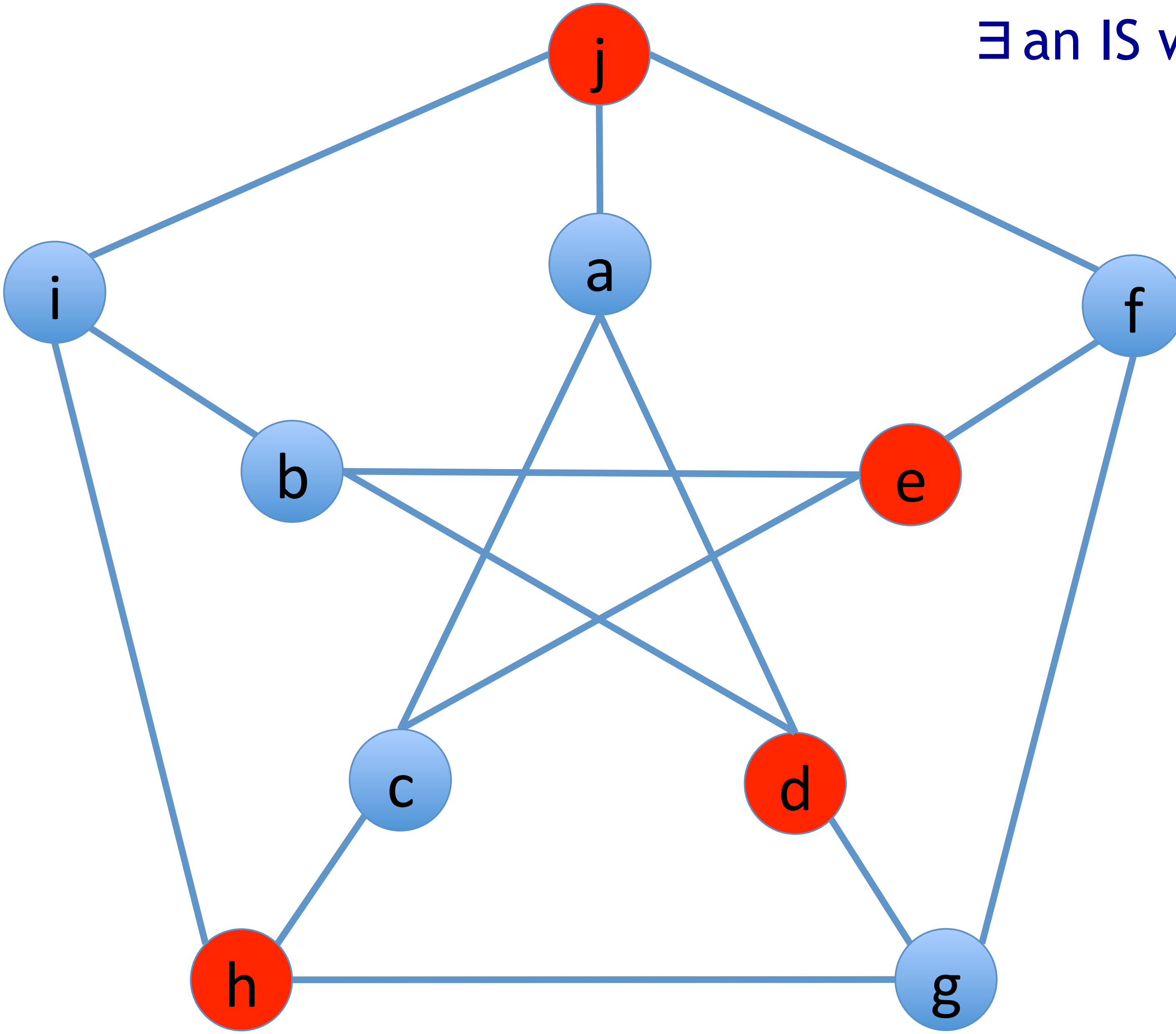
Just take S^C = V-S!

IS $\leq_p$ VC Proof by Picture



IS $\leq_p$ VC Proof by Picture

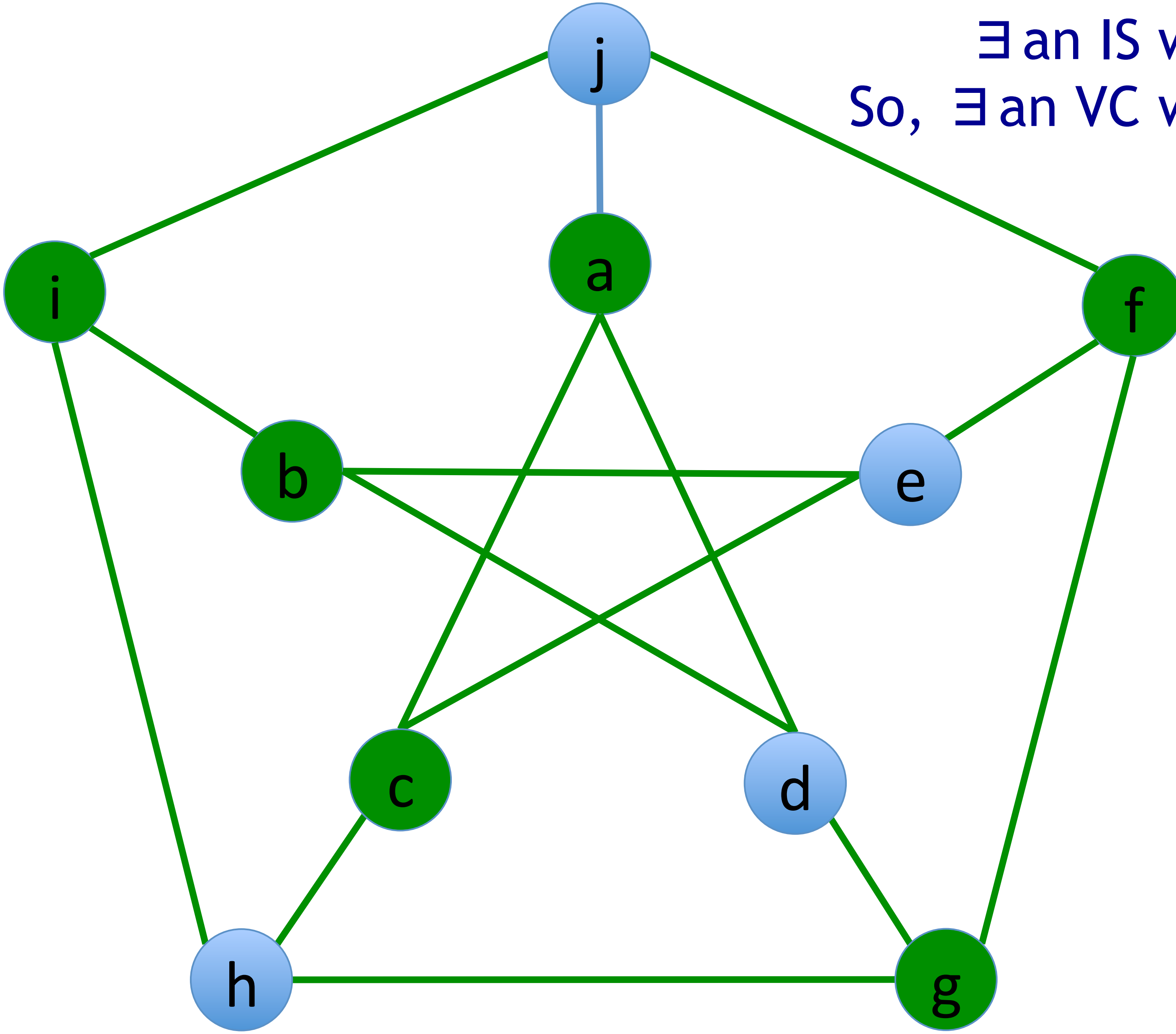
∃ an IS with size 4



IS $\leq_p$ VC Proof by Picture

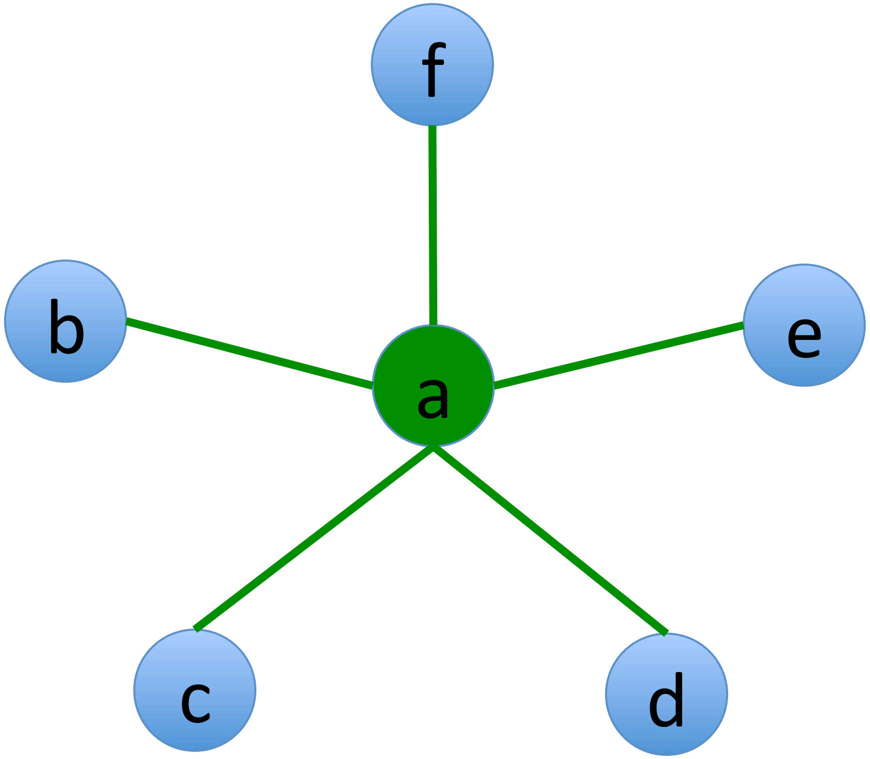
∃ an IS with size 4

So, ∃ an VC with size 6



IS $\leq_p$ VC Proof by Picture (Reverse)

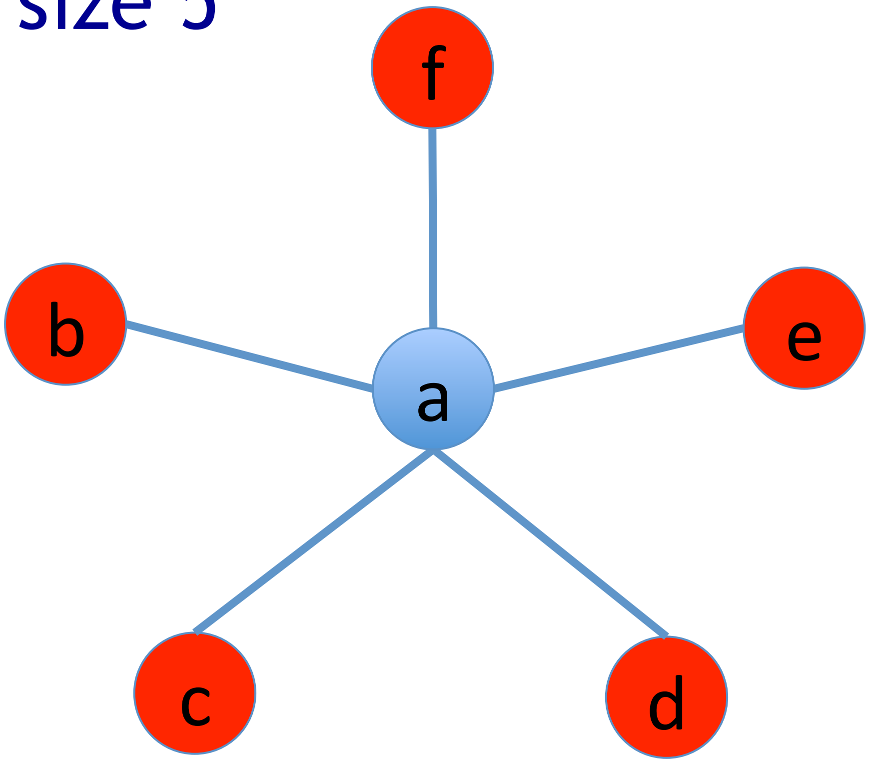
∃ a VS with size 1



IS $\leq_p$ VC Proof by Picture (Reverse)

∃ a VS with size 1

So, ∃ IS with size 5



IS $\leq_p$ VC Proof Idea

In general:

$\exists$ an IS S with size $\geq$ k

iff $\exists$ an VC with size $\leq$ n-k

Just take S^C = V-S!

Q: Runtime of our converter from IS to VC?

$O(1)$

Input to IS: $G(V, E)$, k

Input to VC: $G(V, E)$, $n-k$

Converter only replaces k with $n-k$

IS $\leq_p$ VC

Let X={G(V, E), k} be an instance of IS.

Then convert it to X`={G(V, E), n-k}.

Claim: $\exists$ an IS of size k in G(V, E) iff

$\exists$ a VC of size n-k in G(V, E)

Proof: → let S be an IS s.t. |S| = k

Consider cut (S, S^C=V-S). Since S is an IS

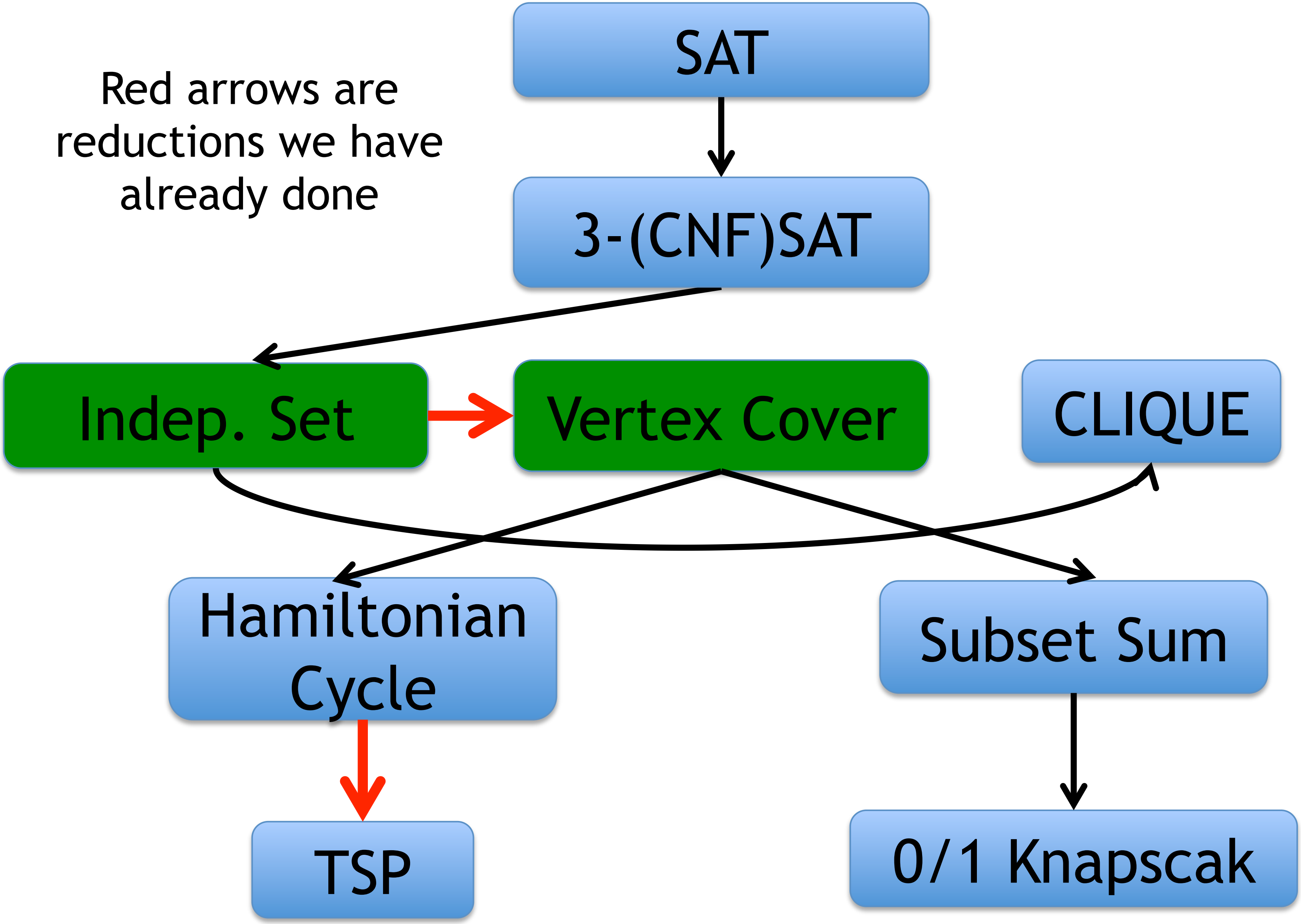
$\forall$(u, v)$\in$E, at least one of u, v both $\in$ V-S or

Therefore S^C is a VC of size n-k

← Is similar (exercise)

Reductions Tree

Red arrows are reductions we have already done



CLIQUE

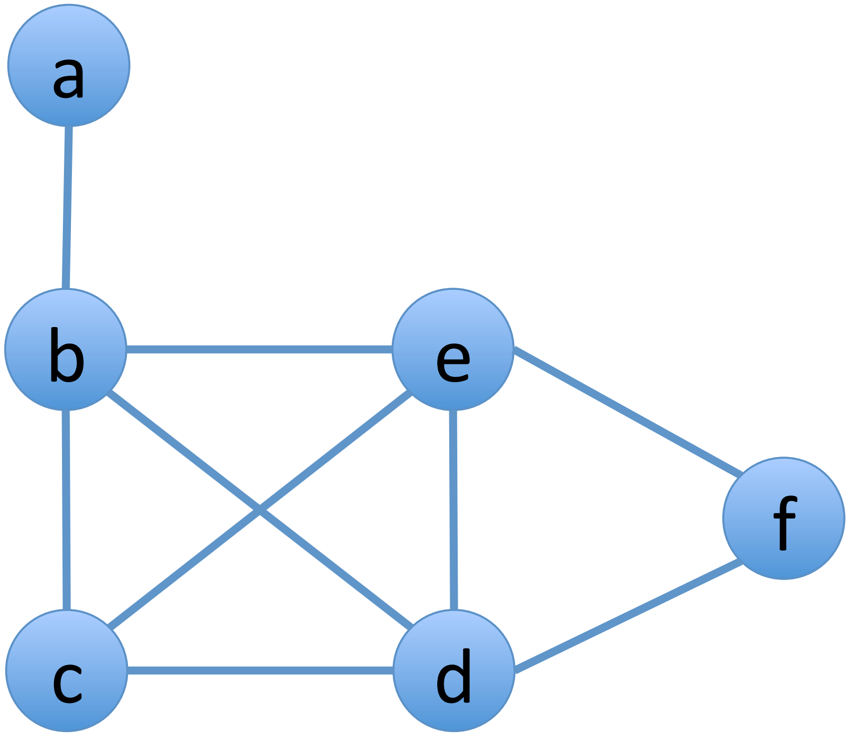
Input: undirected graph $G=(V, E)$ & an integer k

Output: "yes" iff $\exists$ subset $S \subseteq V$ of size $\geq k$ s.t.

$\forall$ u,v s.t u & v both $\in S$: $(u, v) \in E$ i.e.

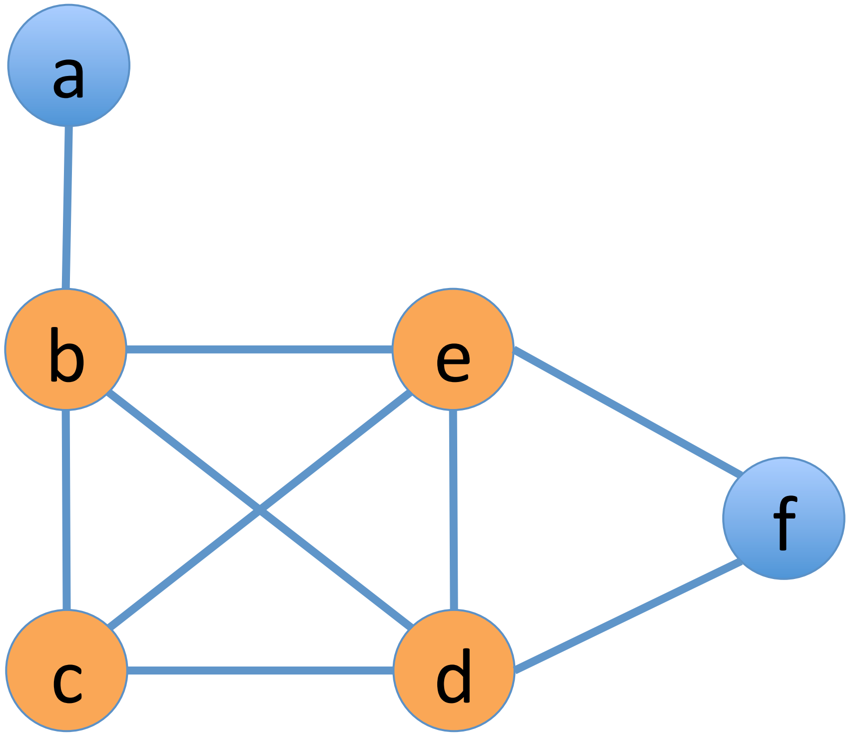
S is a "clique" (all possible edges exist in S)

CLIQUE Example



Q: ∃ an CLIQUE of size ≥ 4?

CLIQUE Example



Q: ∃ an CLIQUE of size ≥ 4?

A: Yes

IS $\leq_p$ CLIQUE Proof Idea

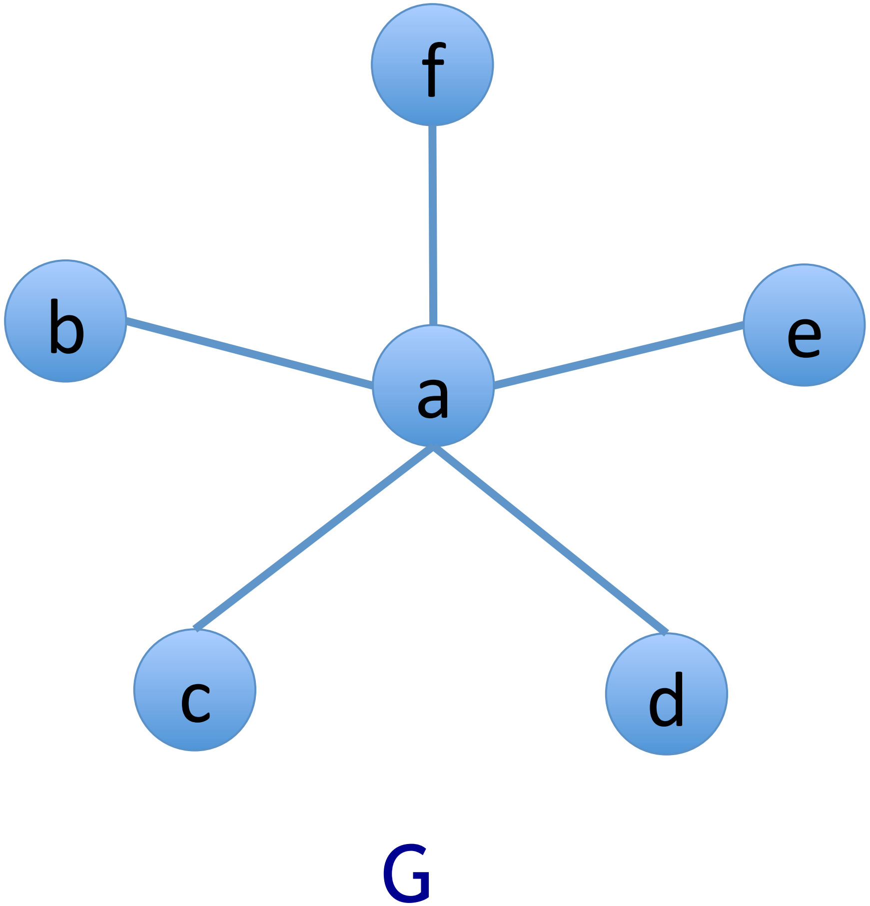
$\exists$ an IS S with size = k in G=(V, E)

iff $\exists$ an CLIQUE with size = k in G^C

(G^C=(V, E^C), contains missing edges of E)

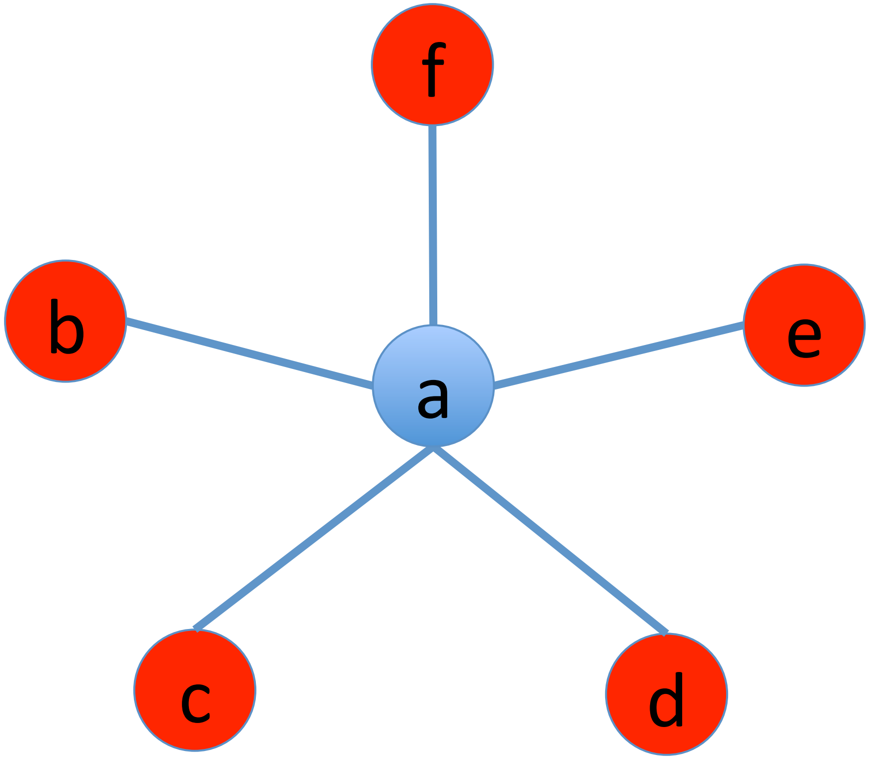
Just take S in G^C!

IS $\leq_p$ CLIQUE Proof by Picture



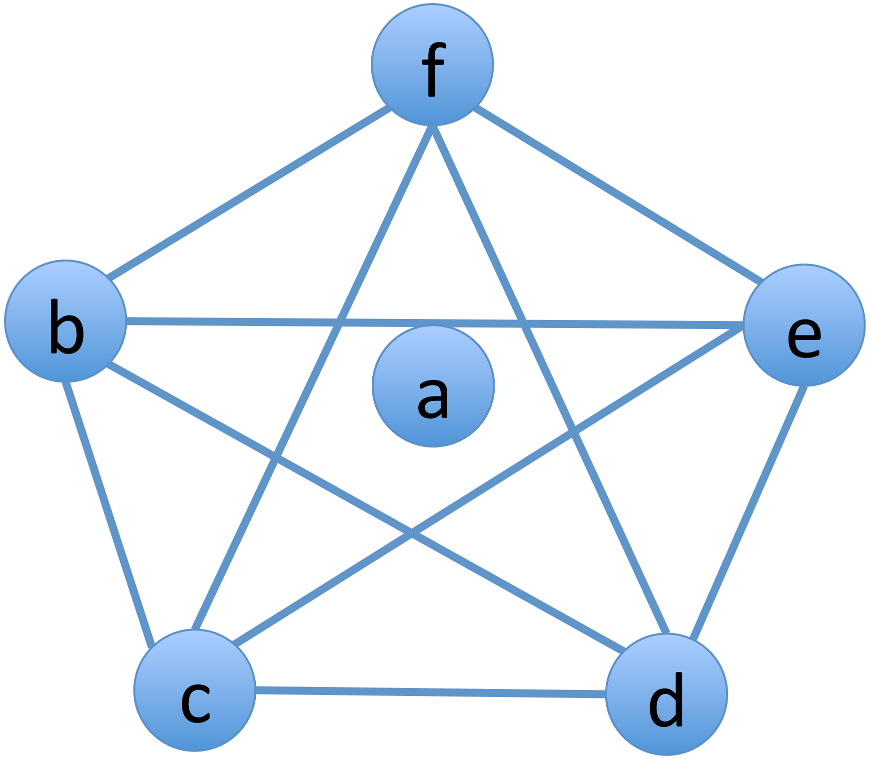
IS $\leq_p$ CLIQUE Proof by Picture

∃ an IS with size 5 in G



G

IS $\leq_p$ CLIQUE Proof by Picture

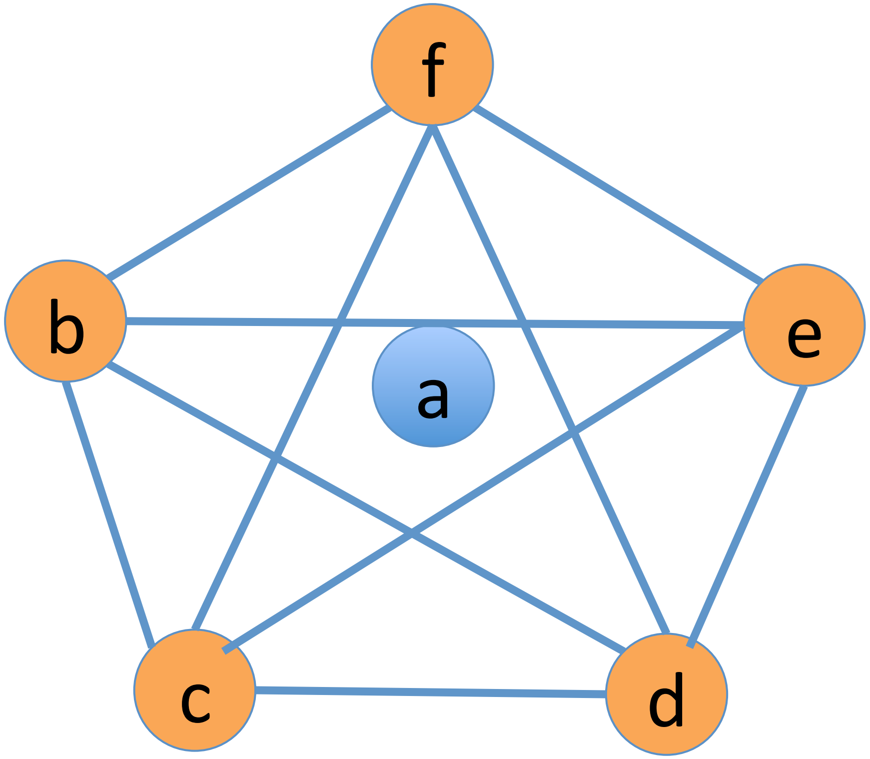


G^C

IS $\leq_p$ CLIQUE Proof by Picture

∃ an IS with size 5 in G

So ∃ a CLIQUE of size 5 in G^C



G^C

IS $\leq_p$ CLIQUE Proof Idea

In general

$\exists$ an IS S with size $\geq$ k in G=(V, E)

iff $\exists$ an CLIQUE with size $\geq$ k in G^C

Q: Runtime of IS to CLIQUE converter?

$O(n^2)$

Input to IS: $G(V, E)$, k

Input to CLIQUE: $G^C(V, E^C)$, k

Converter constructs E^C by adding missing edges

(there may be at most $O(n^2)$ of it)

IS $\leq_p$ CLIQUE

Let X={G(V, E), k} be an instance of IS.

Then convert it to X`={G^C(V, E^C), k}

where E^C is the complement of E

Claim: $\exists$ an IS of size k in G(V, E) iff

$\exists$ a CLIQUE of size k in G^C(V, E^C)

Proof: → let S be an IS s.t. |S| = k

=> $\forall u, v \in S, (u, v) \notin E$

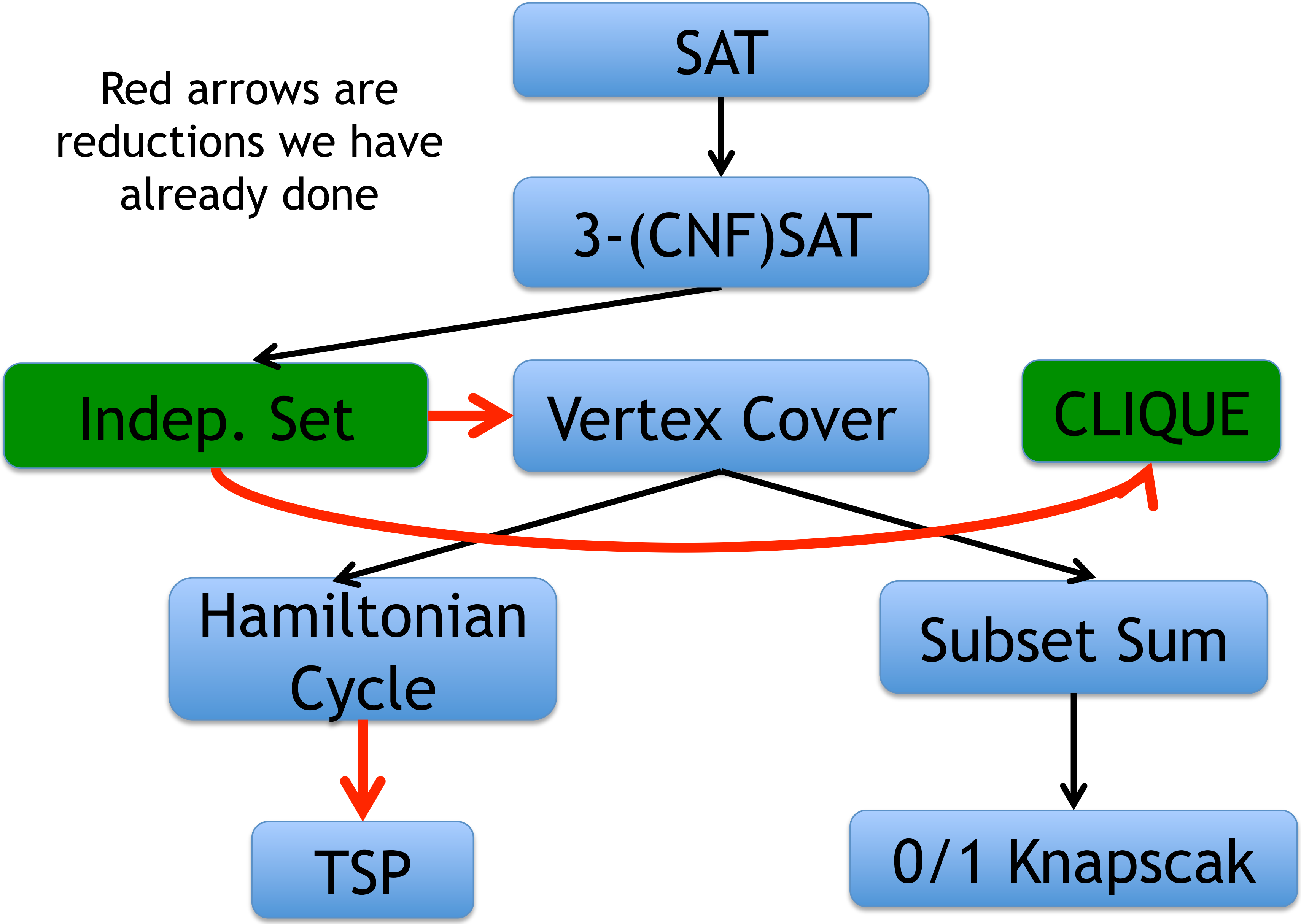
=> $\forall u, v \in S, (u, v) \in E^C$ => S is a clique in G^C

← is similar (exercise)

Important! The reverse argument has to be done!

Reductions Tree

Red arrows are reductions we have already done



3-CNFSAT

Input: A boolean formula φ consisting of:

n boolean variables $x_1, x_2, \ldots, x_n$

m clauses connectives: ∧ (AND), ∨ (OR), ¬ (NOT)

and parantheses s.t.

(1) each clause has 3 distinct literals; AND

(2) φ is in Conjunctive Normal Form

Output: Is φ satisfiable?

Conjunctive Normal Form

x_i are literals

$$\varphi = \boxed{(x_1 \lor \neg x_3 \lor x_7)} \land \boxed{(x_2 \lor \neg x_3 \lor x_5)} \land \boxed{(\neg x_1 \lor x_2 \lor \neg x_7)} \land \ldots$$

Clauses

φ is in CNF: if (1) each clause is an OR of literals or their negations &

(2) φ is an AND of clauses

φ is in 3-CNF: if (1) φ is in CNF &

(2) clauses have 3 distinct literals

3-CNFSAT $\leq_p$ IS

Goal: Convert a 3CNFSAT formula φ into an IS instance (G,k) s.t.

1. Conversion is poly-time; AND
2. IS solution to (G, k) tells us whether φ is satisfiable or not

Ex: $\varphi = (x_1 \lor x_2 \lor x_3) \land (\neg x_1 \lor x_2 \lor x_4) \land$

$(\neg x_2 \lor x_3 \lor x_4) \land (\neg x_1 \lor \neg x_3 \lor \neg x_4)$

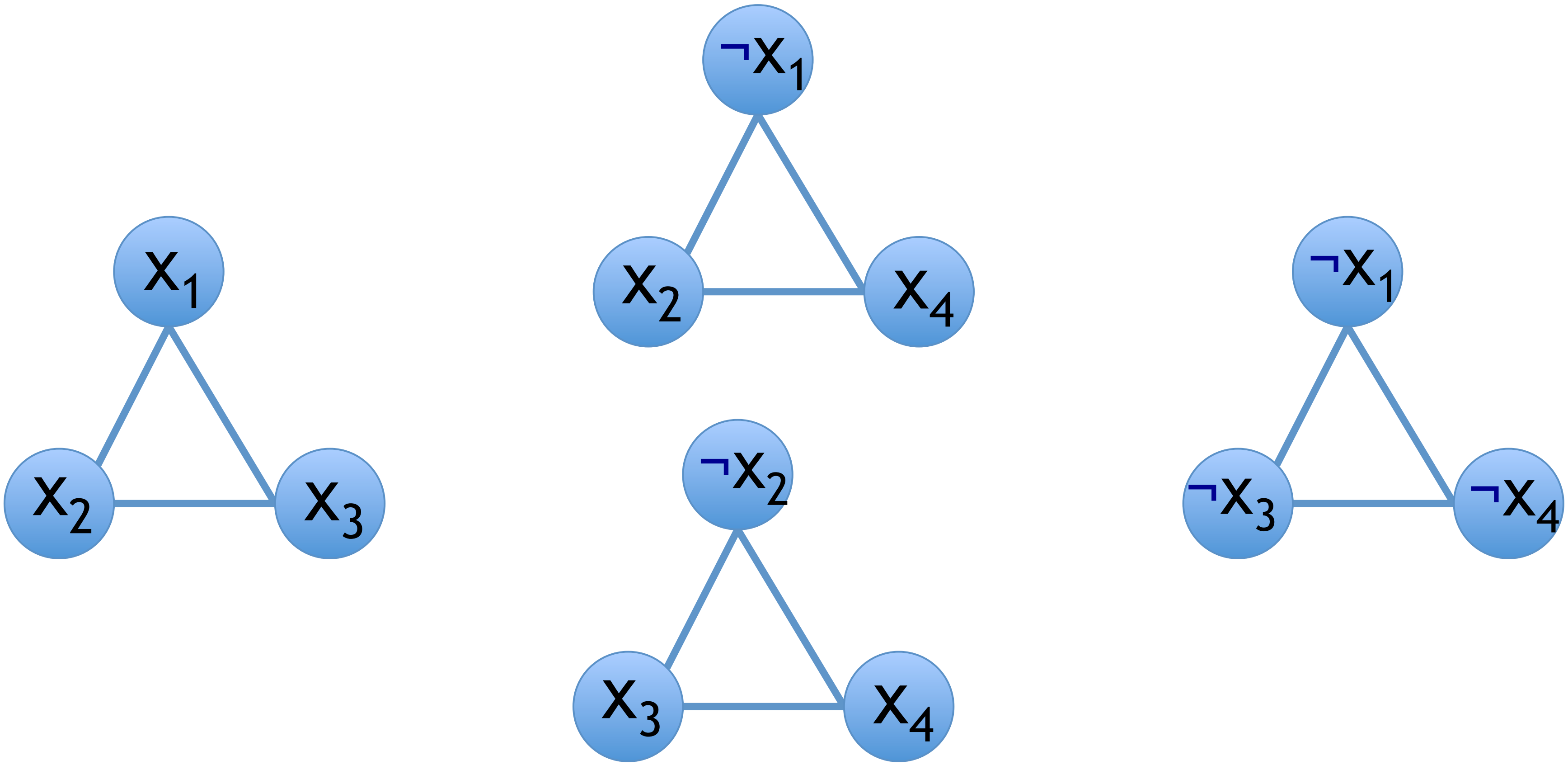
How to convert φ into a graph?

3-CNFSAT to IS Converter Step 1

Ex: $\varphi = (x_1 \lor x_2 \lor x_3) \land (\neg x_1 \lor x_2 \lor x_4) \land$
$(\neg x_2 \lor x_3 \lor x_4) \land (\neg x_1 \lor \neg x_3 \lor \neg x_4)$

For each clause add 3 vertices with the literals as labels; AND

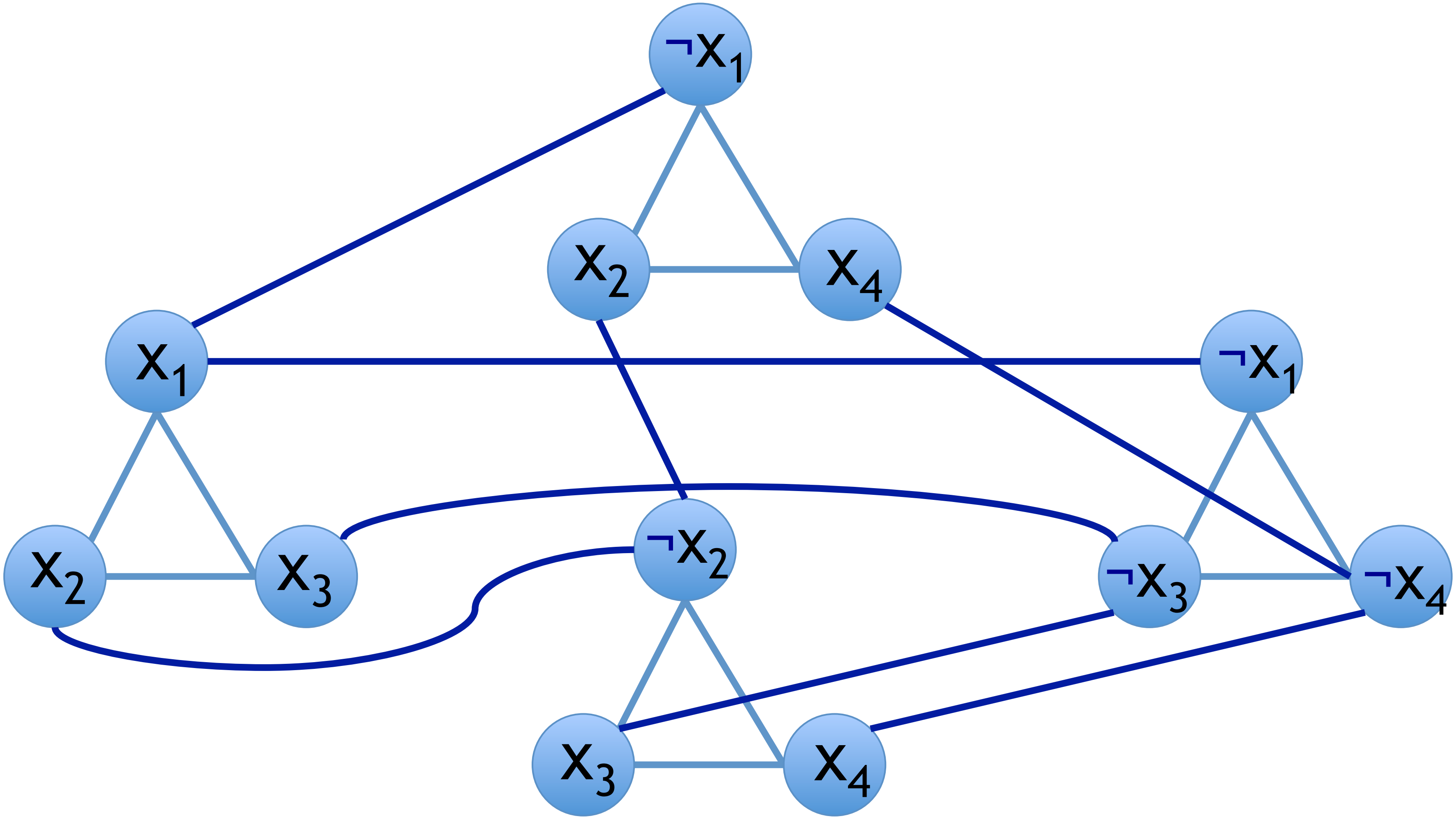
Add each edge between these labels (called "clause gadget")



3-CNFSAT to IS Converter Step 2

Ex: $\varphi = (x_1 \vee x_2 \vee x_3) \wedge (\neg x_1 \vee x_2 \vee x_4) \wedge$
$(\neg x_2 \vee x_3 \vee x_4) \wedge (\neg x_1 \vee \neg x_3 \vee \neg x_4)$

For any two vertices with labels x_i and $\neg x_i$: add another edge



Claim about relation of G and φ

Let m be the # clauses in φ

φ is satisfiable

iff ∃ an IS with size = m in G

Q: Can there be an IS of size > m in G?

A: No, b/c there are m clause gadgets in G

Q: Runtime of 3-CNFSAT to IS converter?

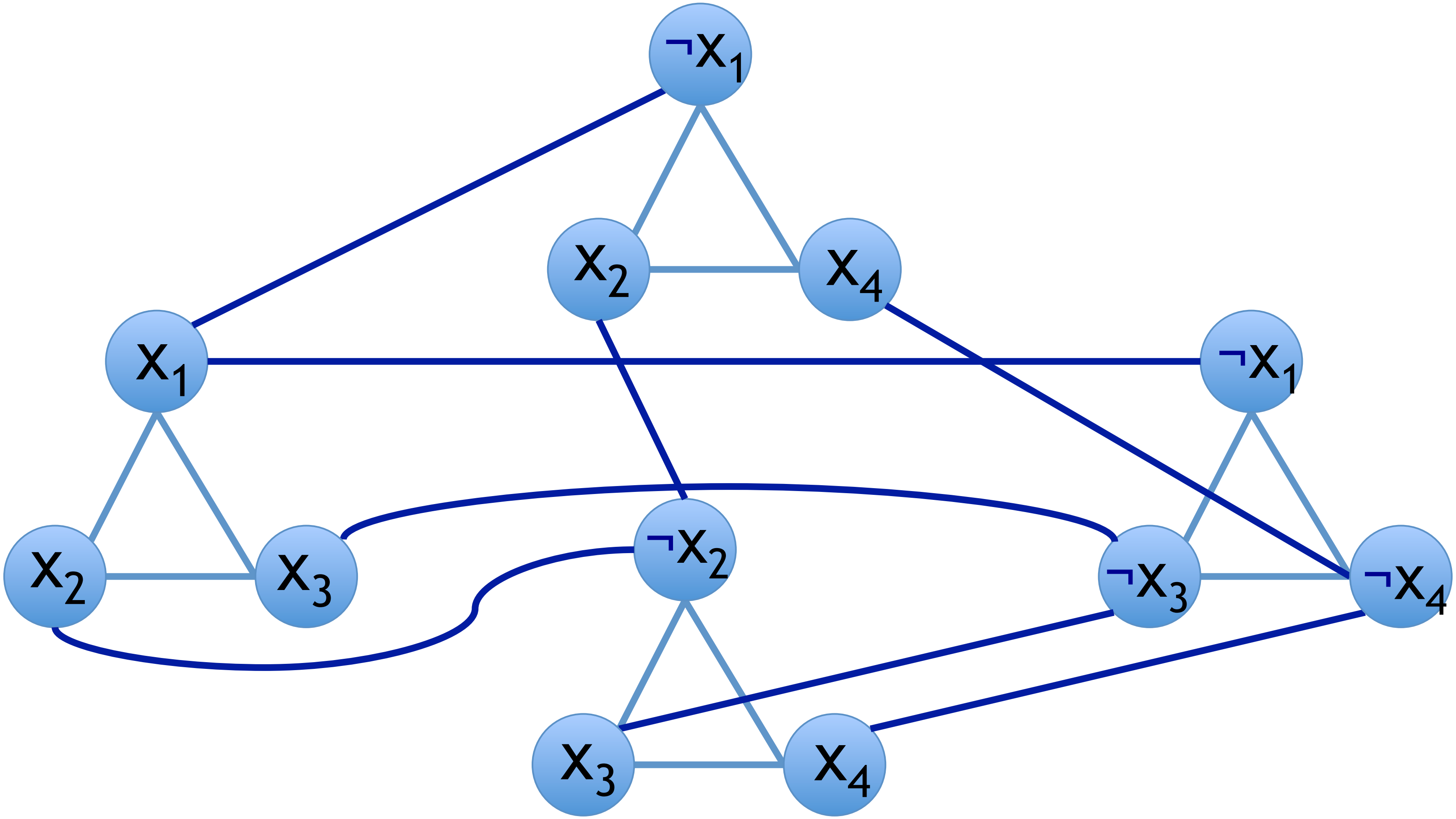
O(poly(m))

Constructing clause gadgets takes O(m) time.

Adding $(x_i, \neg x_i)$ is also poly-time O(mn).

Left Direction: Ex: A=x_1=f; x_2=t; x_3=t; x_4=f

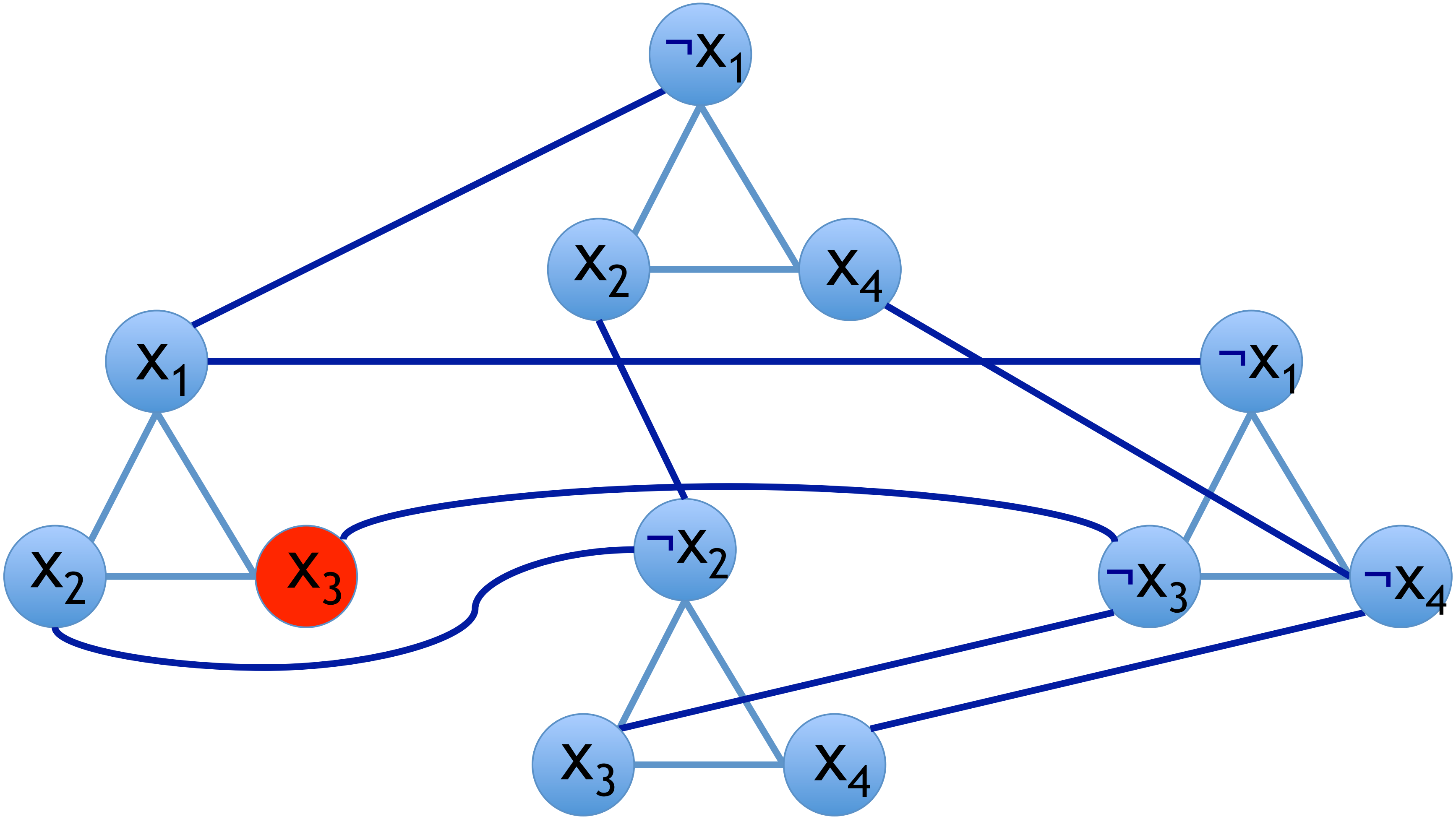
Ex: $\varphi = (x_1 \vee x_2 \vee \boxed{x_3}) \wedge (\boxed{\neg x_1} \vee x_2 \vee x_4) \wedge$
$(\neg x_2 \vee \boxed{x_3} \vee x_4) \wedge (\neg x_1 \vee \neg x_3 \vee \boxed{\neg x_4})$



Left Direction: Ex: A=x_1=f; x_2=t; x_3=t; x_4=f

Ex: $\varphi = (x_1 \lor x_2 \lor \boxed{x_3}) \land (\boxed{\neg x_1} \lor x_2 \lor x_4) \land$
$(\neg x_2 \lor \boxed{x_3} \lor x_4) \land (\neg x_1 \lor \neg x_3 \lor \boxed{\neg x_4})$

For c_1 pick x_3

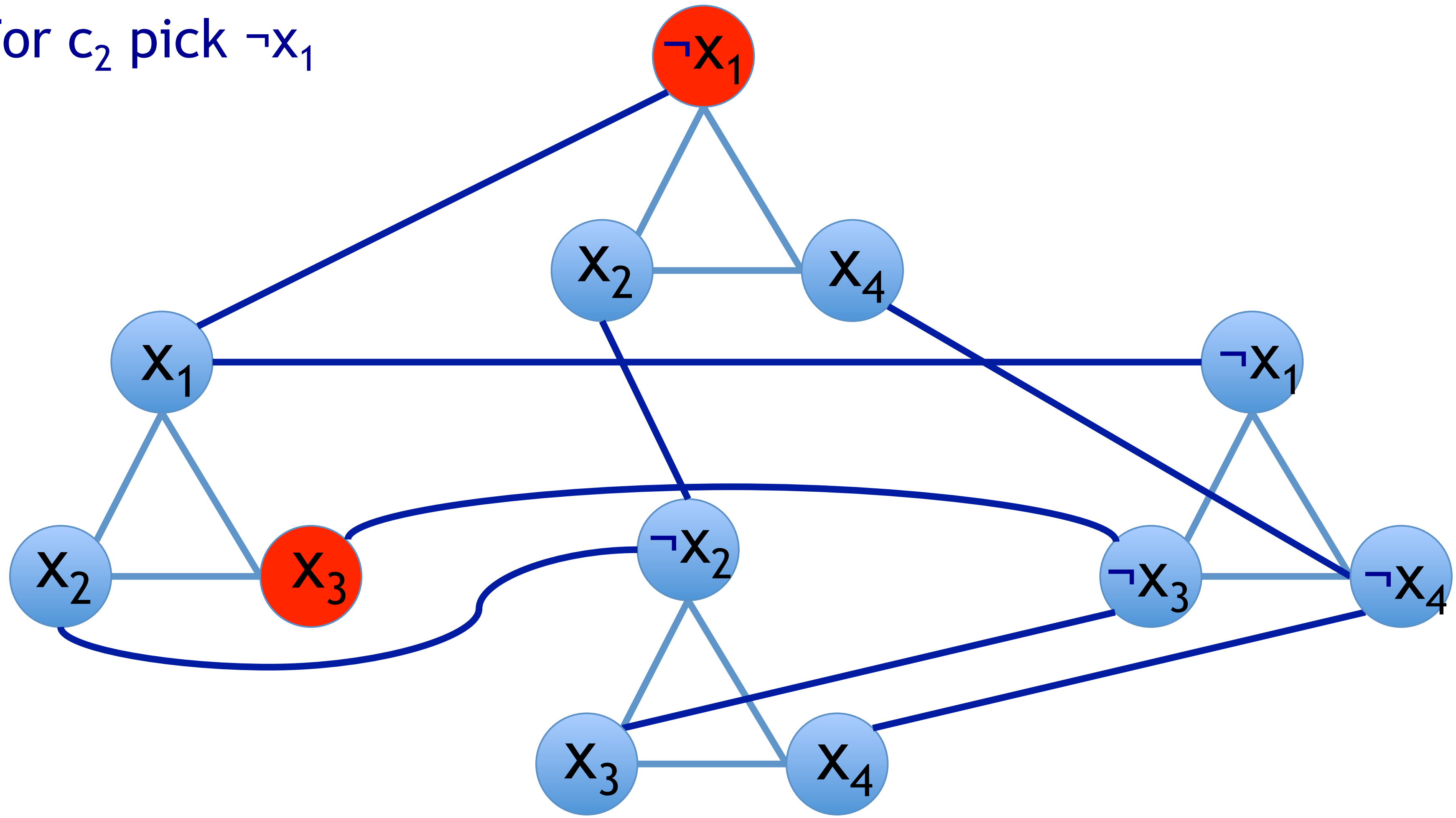


Left Direction: Ex: A=x_1=f; x_2=t; x_3=t; x_4=f

Ex: $\varphi = (x_1 \lor x_2 \lor \boxed{x_3}) \land (\boxed{\neg x_1} \lor x_2 \lor x_4) \land$
$(\neg x_2 \lor \boxed{x_3} \lor x_4) \land (\neg x_1 \lor \neg x_3 \lor \boxed{\neg x_4})$

For c_1 pick x_3

For c_2 pick $\neg x_1$



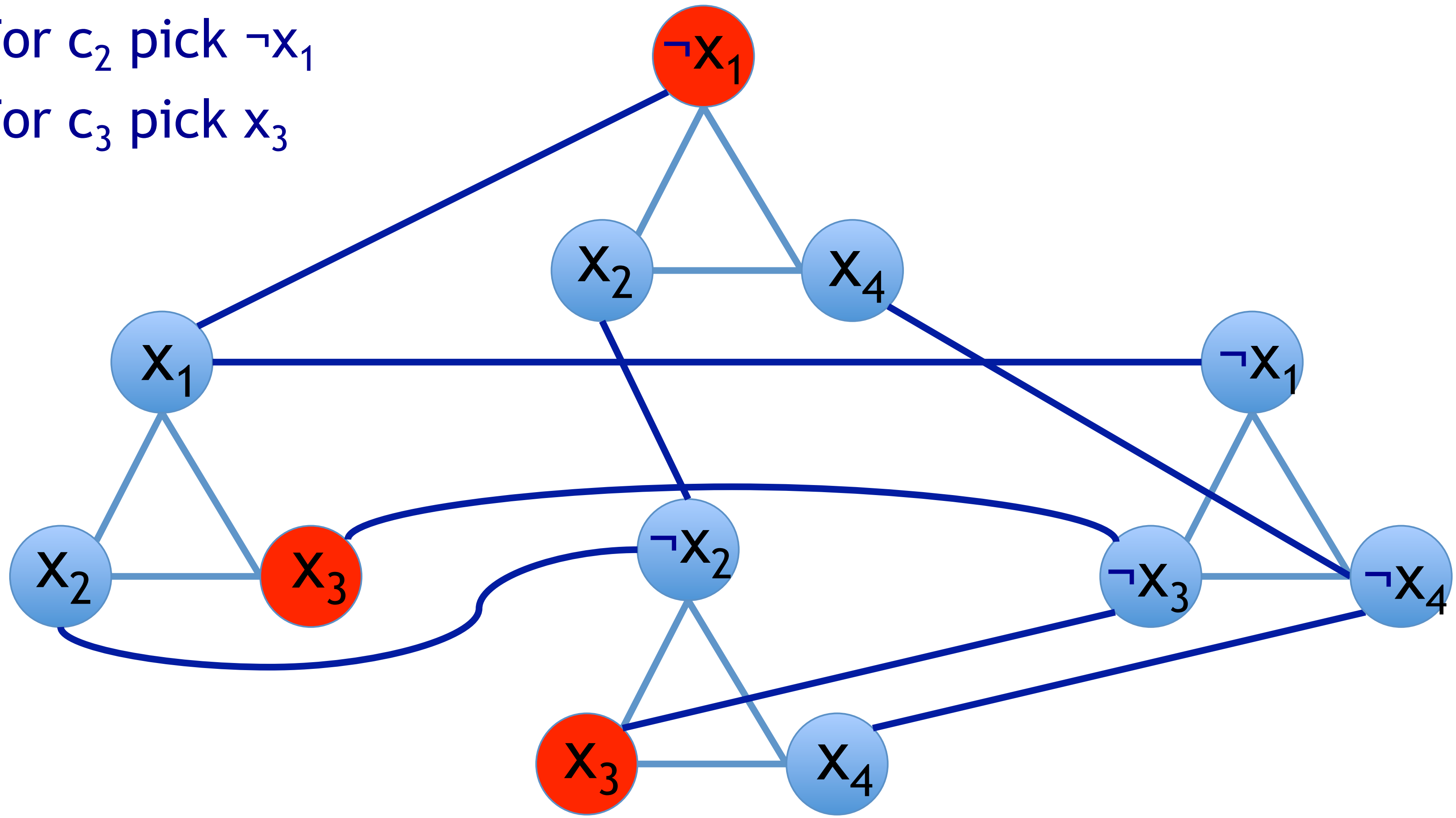
Left Direction: Ex: A=x_1=f; x_2=t; x_3=t; x_4=f

Ex: $\varphi = (x_1 \vee x_2 \vee \boxed{x_3}) \wedge (\boxed{\neg x_1} \vee x_2 \vee x_4) \wedge$
$(\neg x_2 \vee \boxed{x_3} \vee x_4) \wedge (\neg x_1 \vee \neg x_3 \vee \boxed{\neg x_4})$

For c_1 pick x_3

For c_2 pick $\neg x_1$

For c_3 pick x_3



Left Direction: Ex: $A = x_1 = f;\ x_2 = t;\ x_3 = t;\ x_4 = f$

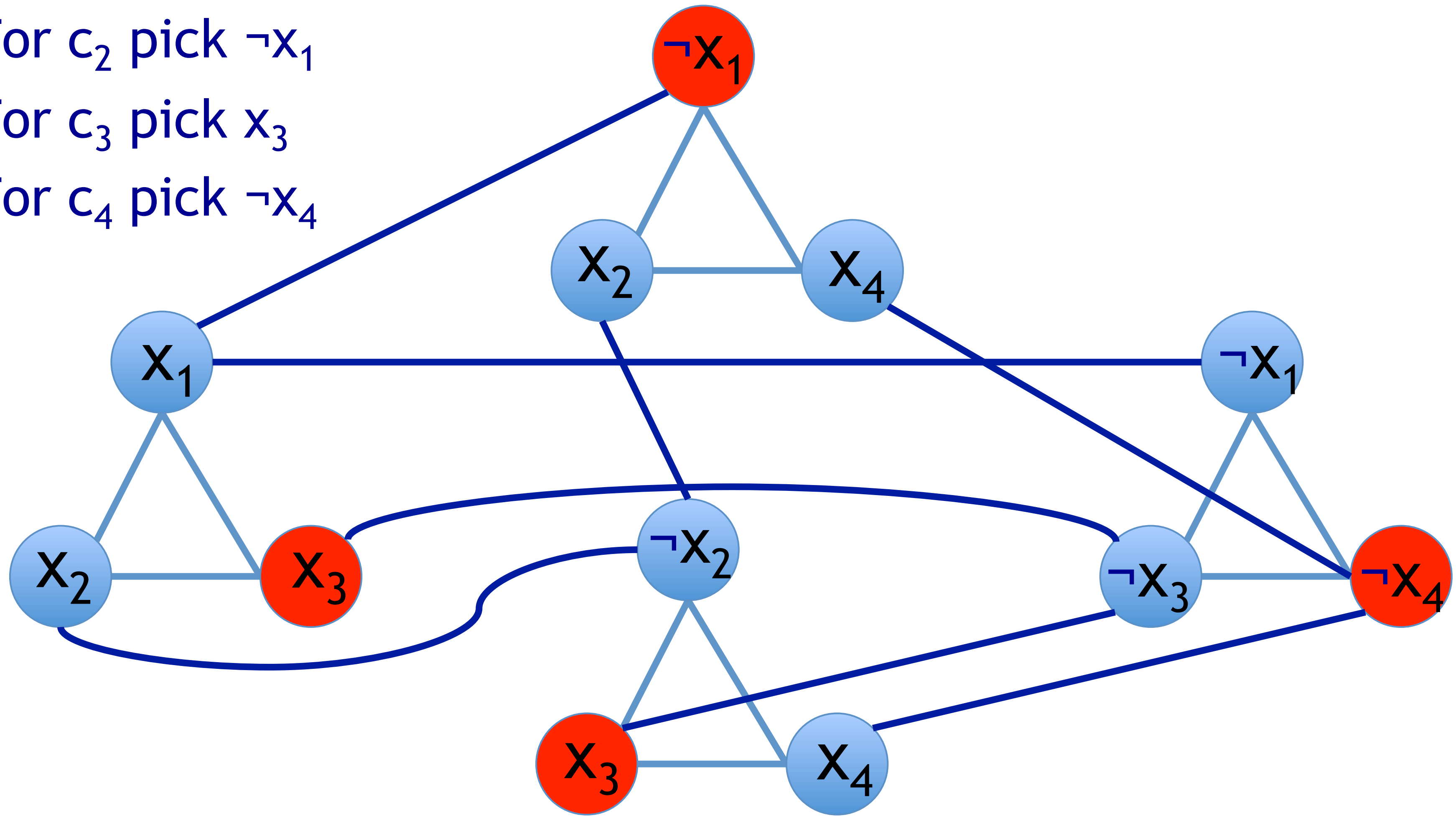
Ex: $\varphi = (x_1 \vee x_2 \vee \boxed{x_3}) \wedge (\boxed{\neg x_1} \vee x_2 \vee x_4) \wedge$
$(\neg x_2 \vee \boxed{x_3} \vee x_4) \wedge (\neg x_1 \vee \neg x_3 \vee \boxed{\neg x_4})$

For c_1 pick x_3

For c_2 pick $\neg x_1$

For c_3 pick x_3

For c_4 pick $\neg x_4$



Left Direction: if φ is sat -> ∃ m-size IS

If φ is satisfiable, i.e., ∃ assignment A satisfying φ

for each clause: ∃ at least one True literal (x_i or $\neg x_j$)

Pick one of those literals arbitrarily in each clause.

Claim: vertices in G of these literals are independent

B/c we picked 1 from each gadget and we cannot

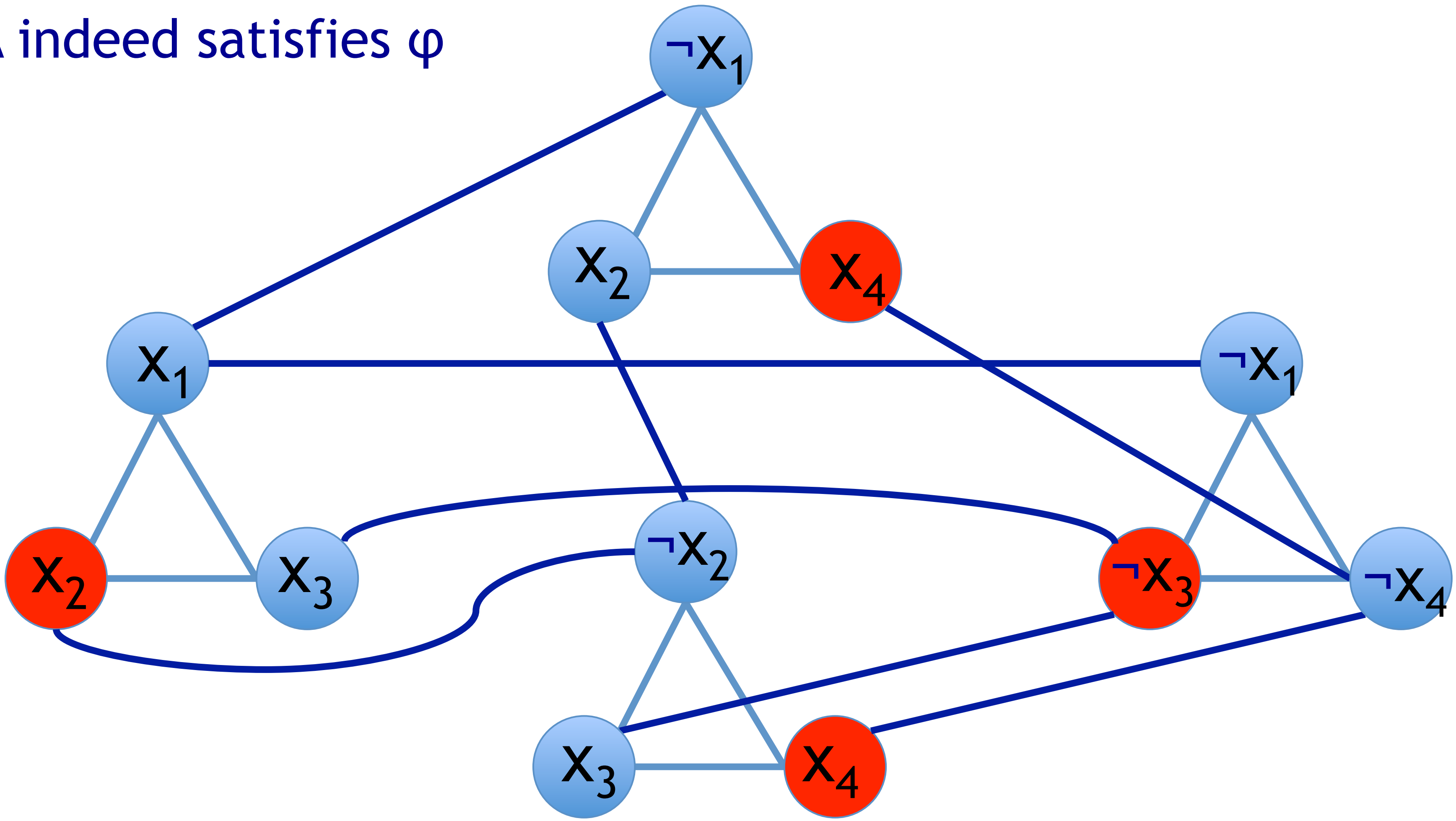
have picked an x_i and $\neg x_i$ at the same time

Right Dir: IS $=\{x_2 \in c_1,\ x_4 \in c_2,\ x_4 \in c_3,\ \neg x_3 \in c_4)$

Ex: $\varphi = (x_1 \vee x_2 \vee x_3) \wedge (\neg x_1 \vee x_2 \vee x_4) \wedge$
$(\neg x_2 \vee x_3 \vee x_4) \wedge (\neg x_1 \vee \neg x_3 \vee \neg x_4)$

$A = x_1 = t/f;\ x_2 = t;\ x_3 = f;\ x_4 = t;$

A indeed satisfies φ



Right Direction: if ∃ m-size IS -> φ is sat

If ∃ m-size IS S in G

=> Each x_i (or $\neg x_i$) ∈ S is in separate clause gadget

(b/c within each gadget all vertices are connected)

⇒ Let **A** be s.t. we set each x_i (or $\neg x_i$) ∈ S to True

Note we cannot set x_i and $\neg x_i$ to True simultaneously

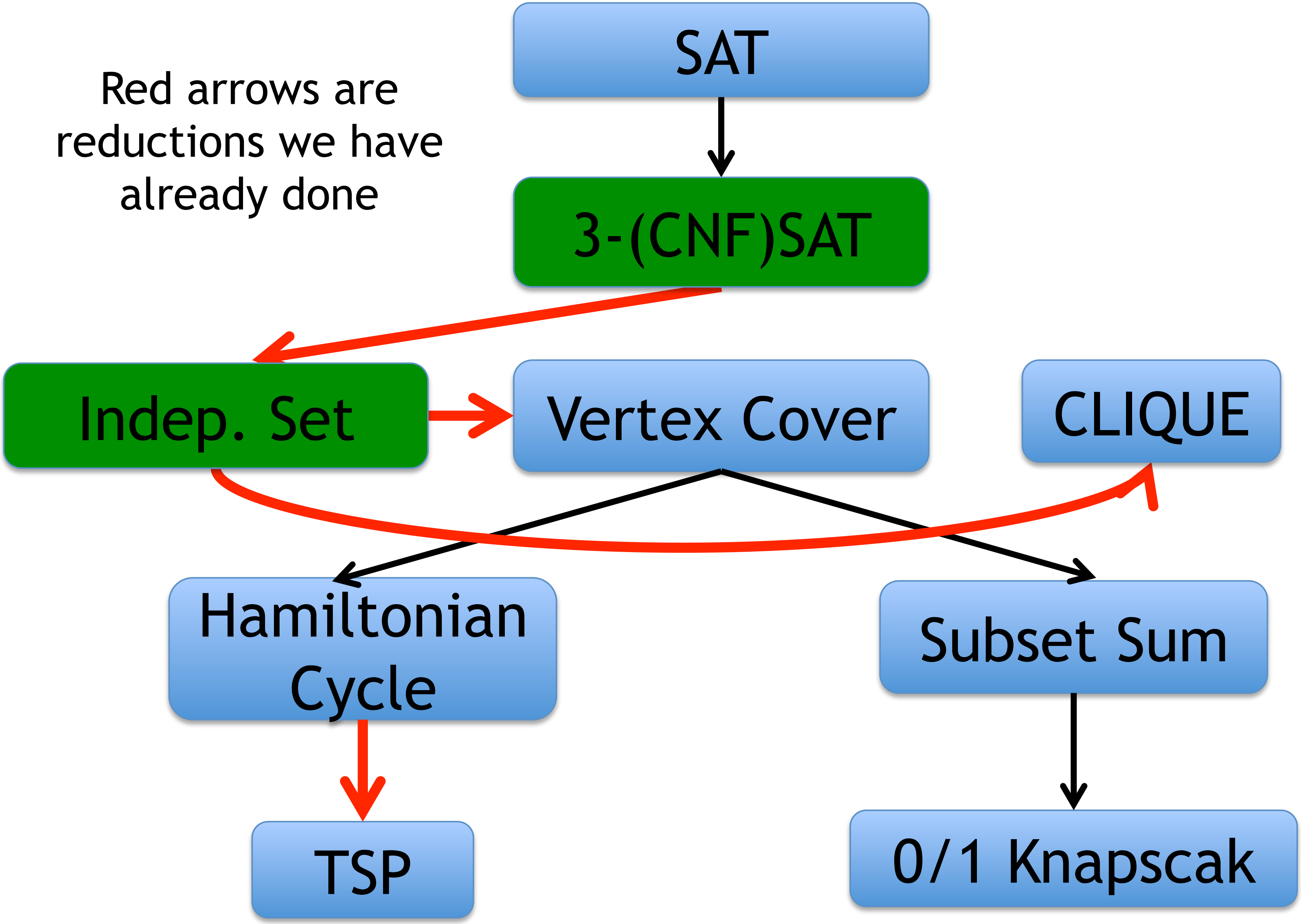
b/c in G, there is an edge between each (x_i, $\neg x_i$).

⇒ Assign non-assigned literals x_i (or $\neg x_i$) arbitrarily

⇒ Claim: **A** satisfies φ b/c by construction there is at least one literal in each clause that's T

Reductions Tree

Red arrows are reductions we have already done



Subset Sum

Input: A set of X:{x_1, x_2, …, x_n} of integers and a target t

Output: YES if $\exists S \subseteq X$ s.t sum of S's elements equals exactly t

Example:

X = {1, 4, 16, 64, 256, 1040, 1041, 1093, 1284, 1344}

t = 3754

YES: S = {1, 16, 64, 1040, 1093, 1284}

Subset Sum $\leq_p$ 0-1 Knapsack

Goal: Take Subset Sum instance $\{x_1, \ldots, x_n\}$, t

Turn into a 0-1 Knapsack Instance:

$A=\{v_1, \ldots, v_n\}$, $B=\{w_1, \ldots, w_n\}$, W

Note: 0-1 Knapsack-DECISION: n items, W, V

$\exists$ a set of items with weight $\leq W$

and value $\geq V$

Idea: $A=\{x_1, \ldots, x_n\}$, $B=\{x_1, \ldots, x_n\}$, $W=t$, $V=t$

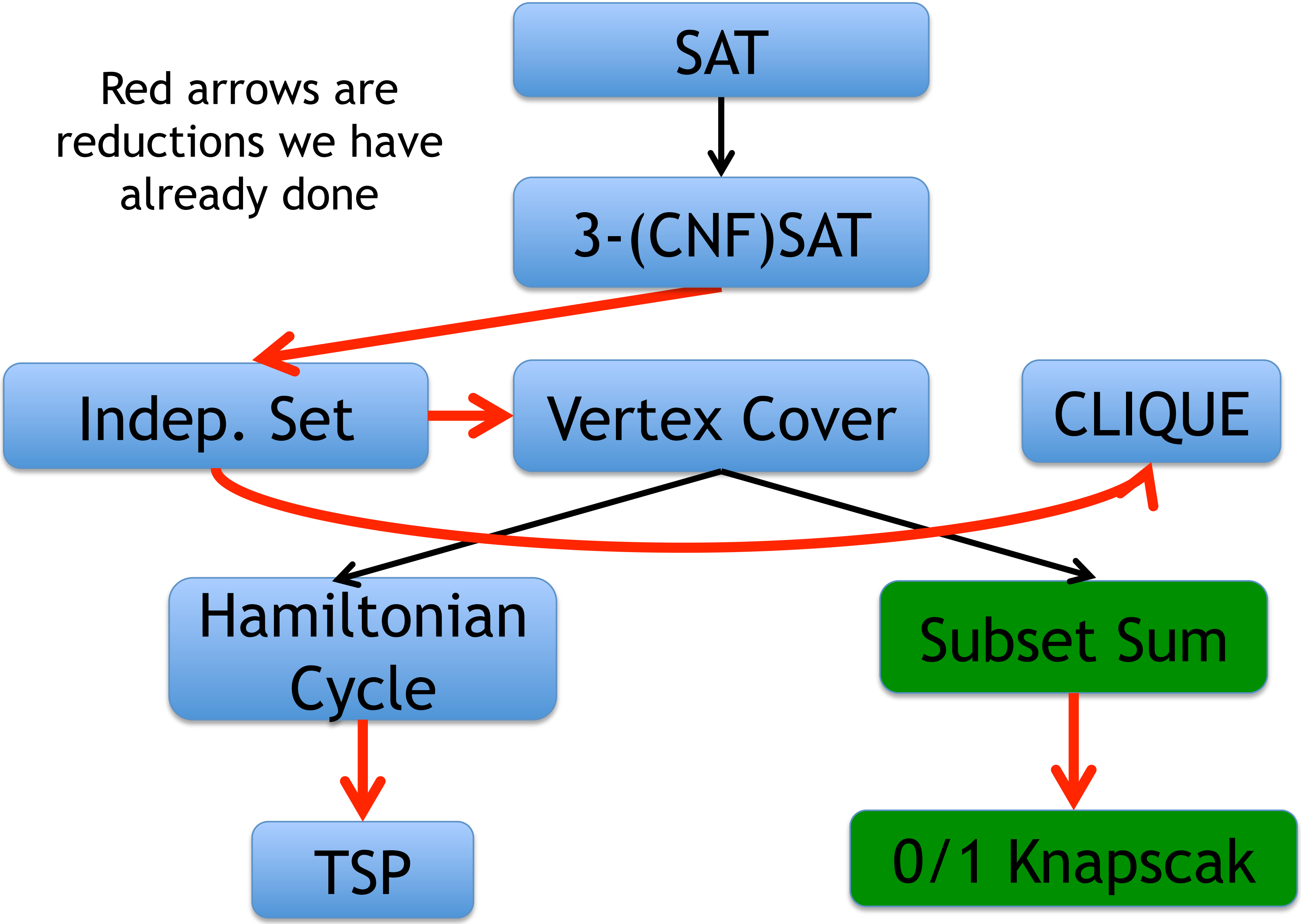
Make each item s.t 1 weight always equal 1 value.

Ask if we can pack into a knapsack of size t, value at least t

Note value can't be > t because each weight has 1 value

Reductions Tree

Red arrows are reductions we have already done



Recall Vertex Cover (VC)

Input: undirected graph $G=(V, E)$ & an integer k

Output: "yes" iff $\exists$ subset $S \subseteq V$ of size $\leq k$ s.t.

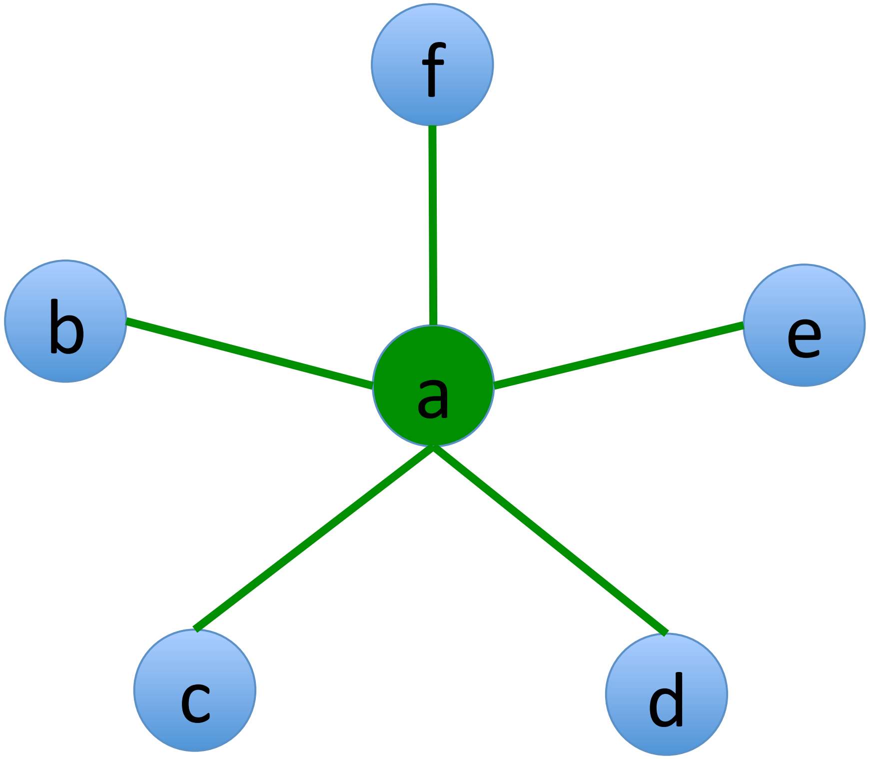
$\forall (u, v) \in E$, either $u \in S$ or $v \in S$

(each edge is "covered" by at least one vertex $\in S$)

VC Example

Q: ∃ a VS with size ≤ 1?

A: Yes



Vertex Cover $\leq_p$ Subset Sum

Goal: Take VC instance G, k

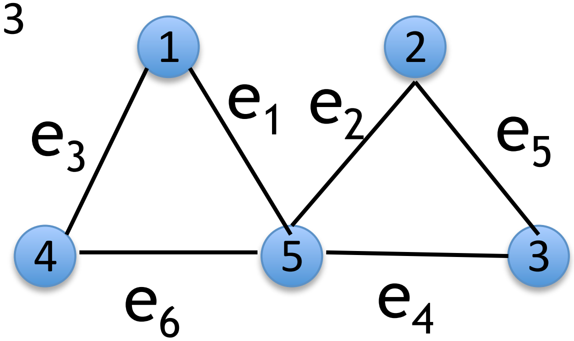
Turn into a Subset Sum Instance:

$X=\{x_1, \ldots, x_n\}$, t s.t

$\exists$ a VC of size $\leq k \leftrightarrow \exists$ $S \subseteq X$ s.t. sum of S equal exactly t

Vertex Cover $\leq_p$ Subset Sum

k = 3



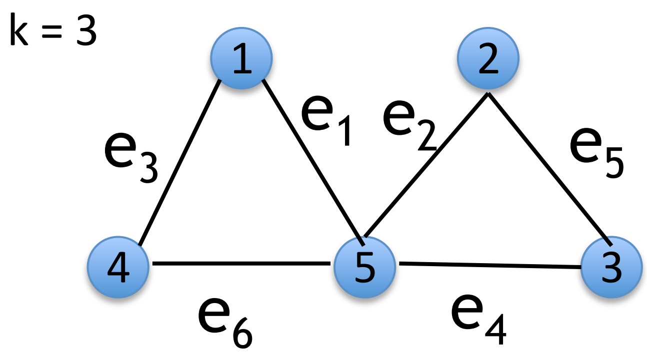
<u>Interpretations:</u>

v_i are vertices

y_i will be “place holders”

		e_1	e_2	e_3	e_4	e_5	e_6	decimal
v_1	1	1	0	1	0	0	0	5184
v_2	1	0	1	0	0	1	0	4356
v_3	1	0	0	0	1	1	0	4116
v_4	1	0	0	1	0	0	1	4161
v_5	1	1	1	0	1	0	1	5393
y_1	0	1	0	0	0	0	0	1024
y_2	0	0	1	0	0	0	0	256
y_3	0	0	0	1	0	0	0	64
y_4	0	0	0	0	1	0	0	16
y_5	0	0	0	0	0	1	0	4
y_6	0	0	0	0	0	0	1	1
t	3	2	2	2	2	2	2	15018

Vertex Cover $\leq_p$ Subset Sum



<u>Interpretations:</u>

v_i are vertices

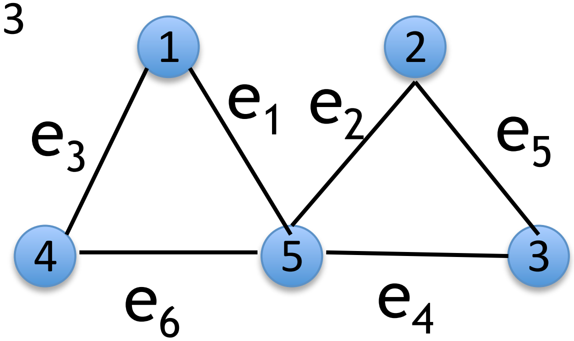
y_i will be "place holders"

1st Clmn: will force to select exactly k items

		e_1	e_2	e_3	e_4	e_5	e_6	decimal
v_1	1	1	0	1	0	0	0	5184
v_2	1	0	1	0	0	1	0	4356
v_3	1	0	0	0	1	1	0	4116
v_4	1	0	0	1	0	0	1	4161
v_5	1	1	1	0	1	0	1	5393
y_1	0	1	0	0	0	0	0	1024
y_2	0	0	1	0	0	0	0	256
y_3	0	0	0	1	0	0	0	64
y_4	0	0	0	0	1	0	0	16
y_5	0	0	0	0	0	1	0	4
y_6	0	0	0	0	0	0	1	1
t	3	2	2	2	2	2	2	15018

Vertex Cover $\leq_p$ Subset Sum

k = 3



<u>Interpretations:</u>

v_i are vertices

y_i will be "place holders"

1^{st} Clmn: will force to select exactly k items

each v_i row: adjacent edges of v_i

Interpret numbers as base k+1 (in example = 4)

		e_1	e_2	e_3	e_4	e_5	e_6	decimal
v_1	1	1	0	1	0	0	0	5184
v_2	1	0	1	0	0	1	0	4356
v_3	1	0	0	0	1	1	0	4116
v_4	1	0	0	1	0	0	1	4161
v_5	1	1	1	0	1	0	1	5393
y_1	0	1	0	0	0	0	0	1024
y_2	0	0	1	0	0	0	0	256
y_3	0	0	0	1	0	0	0	64
y_4	0	0	0	0	1	0	0	16
y_5	0	0	0	0	0	1	0	4
y_6	0	0	0	0	0	0	1	1
t	3	2	2	2	2	2	2	15018

Claim

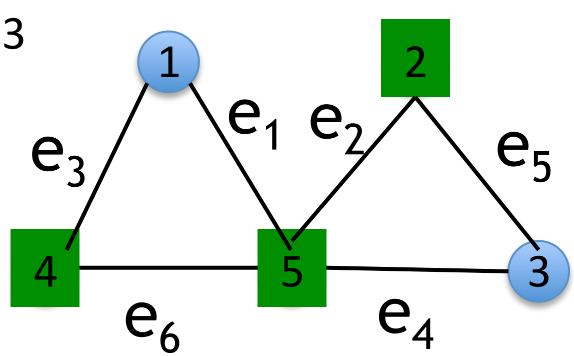
$\exists$ *VC C of size* $\leq k \leftrightarrow \exists\ S \subseteq X$ *with sum 15018*

$\exists$ VC C of size $\leq k \rightarrow \exists$ S $\subseteq$ X with sum 15018

1) Complete C to size exactly k by adding any k-|C| vertices.

2) Fix missing digits by adding Y rows

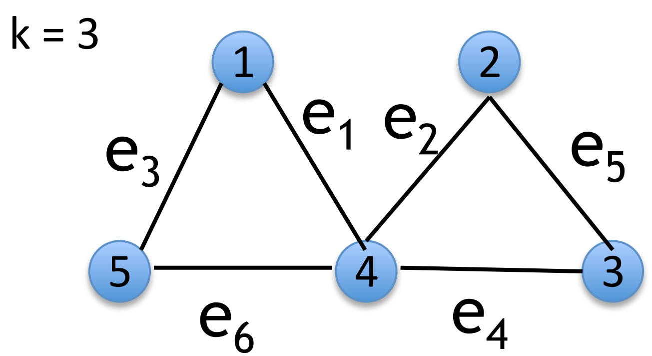
{4356, 4161, 5393, 1024, 64, 16, 4} add up to 15018



		e_1	e_2	e_3	e_4	e_5	e_6	decimal
v_1	1	1	0	1	0	0	0	5184
v_2	1	0	1	0	0	1	0	4356
v_3	1	0	0	0	1	1	0	4116
v_4	1	0	0	1	0	0	1	4161
v_5	1	1	1	0	1	0	1	5393
y_1	0	1	0	0	0	0	0	1024
y_2	0	0	1	0	0	0	0	256
y_3	0	0	0	1	0	0	0	64
y_4	0	0	0	0	1	0	0	16
y_5	0	0	0	0	0	1	0	4
y_6	0	0	0	0	0	0	1	1
t	3	2	2	2	2	2	2	15018

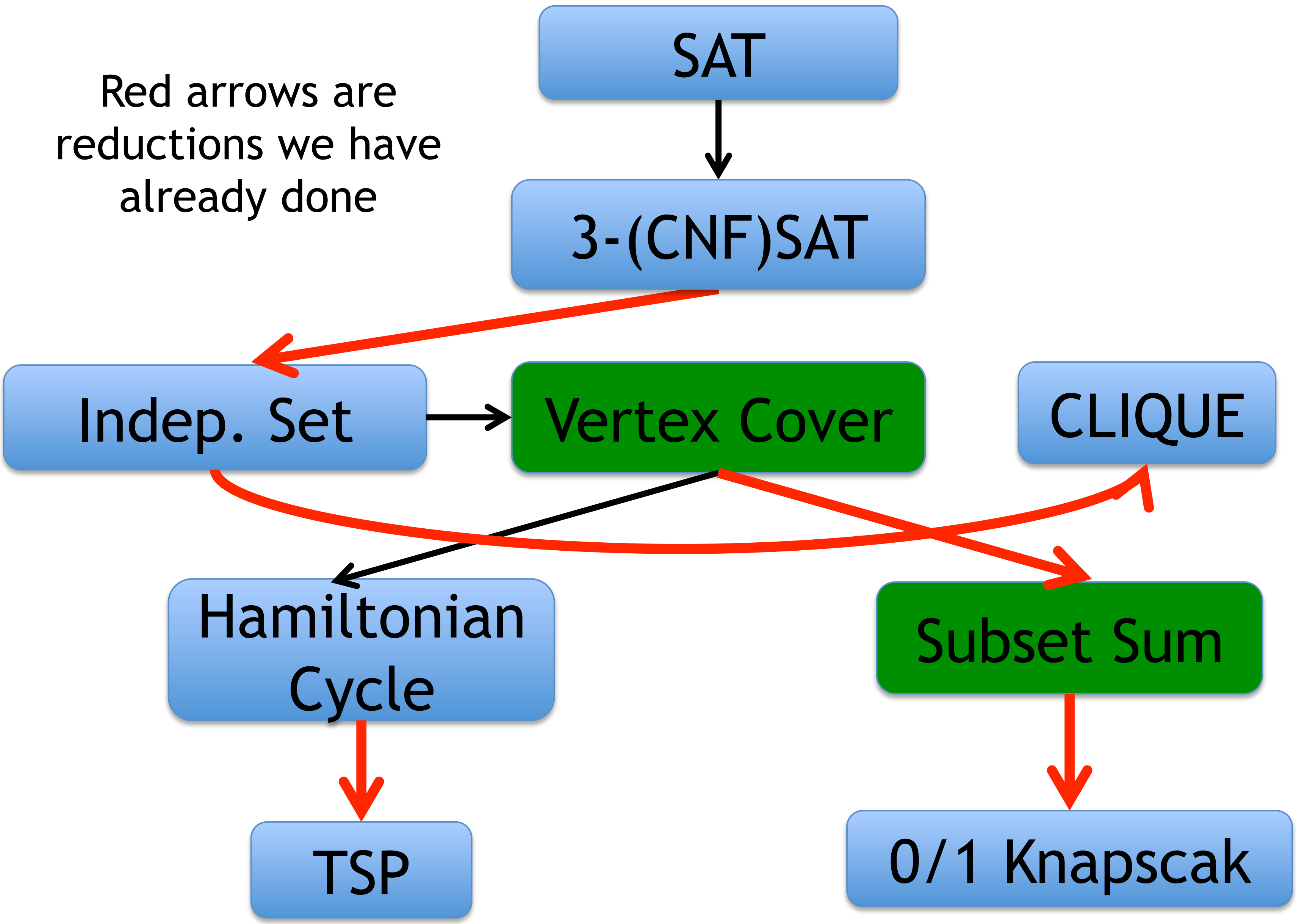
$\exists$ VC C of size $\leq$k $\leftarrow \exists$ S $\subseteq$ X with sum 15018

Suppose S sums to t.
Let C = S $\cap$ V (red rows)
1) each e_i has 3 1's, so no carry overs
2) |C| = k (b/c the first digit is k
3) C is a VC b/c at least one $v_j \in$ C must contribute a 1 to each "column" e_i, i.e., covers e_i.



		e_1	e_2	e_3	e_4	e_5	e_6	decimal
v_1	1	1	0	1	0	0	0	5184
v_2	1	0	1	0	0	1	0	4356
v_3	1	0	0	0	1	1	0	4116
v_4	1	0	0	1	0	0	1	4161
v_5	1	1	1	0	1	0	1	5393
y_1	0	1	0	0	0	0	0	1024
y_2	0	0	1	0	0	0	0	256
y_3	0	0	0	1	0	0	0	64
y_4	0	0	0	0	1	0	0	16
y_5	0	0	0	0	0	1	0	4
y_6	0	0	0	0	0	0	1	1
t	$\boxed{3}$	2	2	2	2	2	2	15018

Previous Reductions Tree



Your Problem is NP-complete. Now What?

- Option 1: Focus to special-case inputs.
  - Ex: Independent Set is NP-complete.
  - Focusing on line graphs, had a O(n) DP alg.
- Option 2: Find an approximate answer.
  - Will show a very simple algorithm for 0-1 Knapsack.
- Option 3: Be exponential time but better than brute-force search.
  - 0-1 Knapscak O(nW) runtime DP algorihm.
- Option 4: Heuristics: fast algorithms that are not always correct (or even approximate)
- Option 5: Mix some of these options

Dealing With NP-complete Problems

For NP-complete Problems the algorithmic tools in our toolbox can be used as is.

But we have to give up something: (1) generality, (2) exactness, or (3) efficiency.