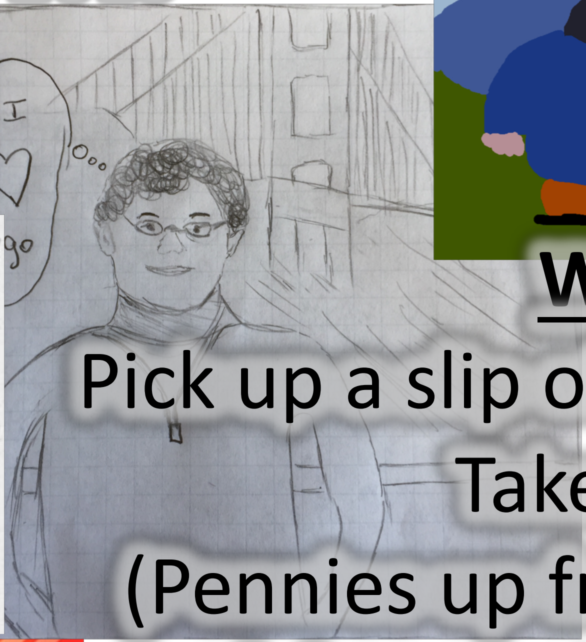
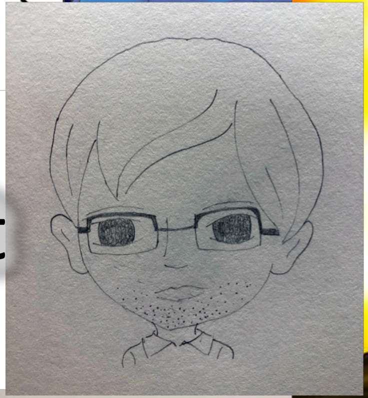
CS4102 Algorithms

Spring 2019





Warm up:

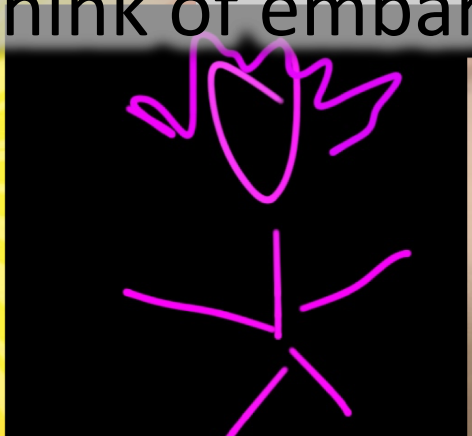
Pick up a slip of paper from the front

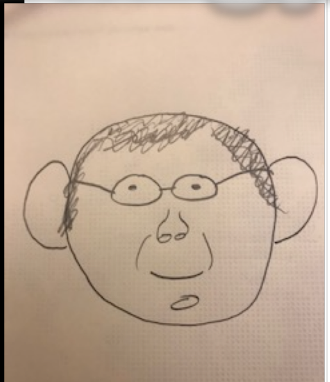
Take out a coin

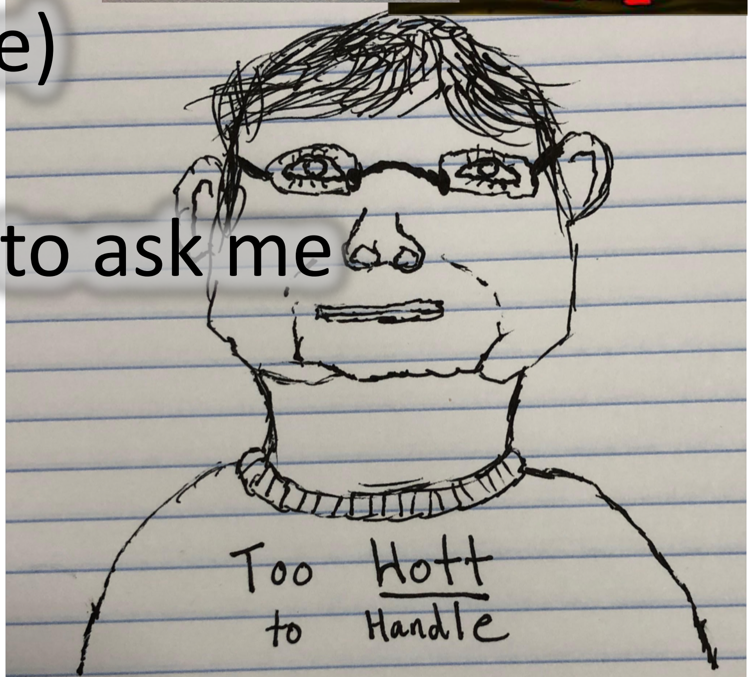
(Pennies up front if you need one)

(please return them at end)

Think of embarrassing yes/no questions to ask me







Today’s Keywords

- Reductions
- NP-Completeness
- Vertex Cover
- Independent Set
- 3-SAT
- Clique
- Differential Privacy

CLRS Readings

- Chapter 34

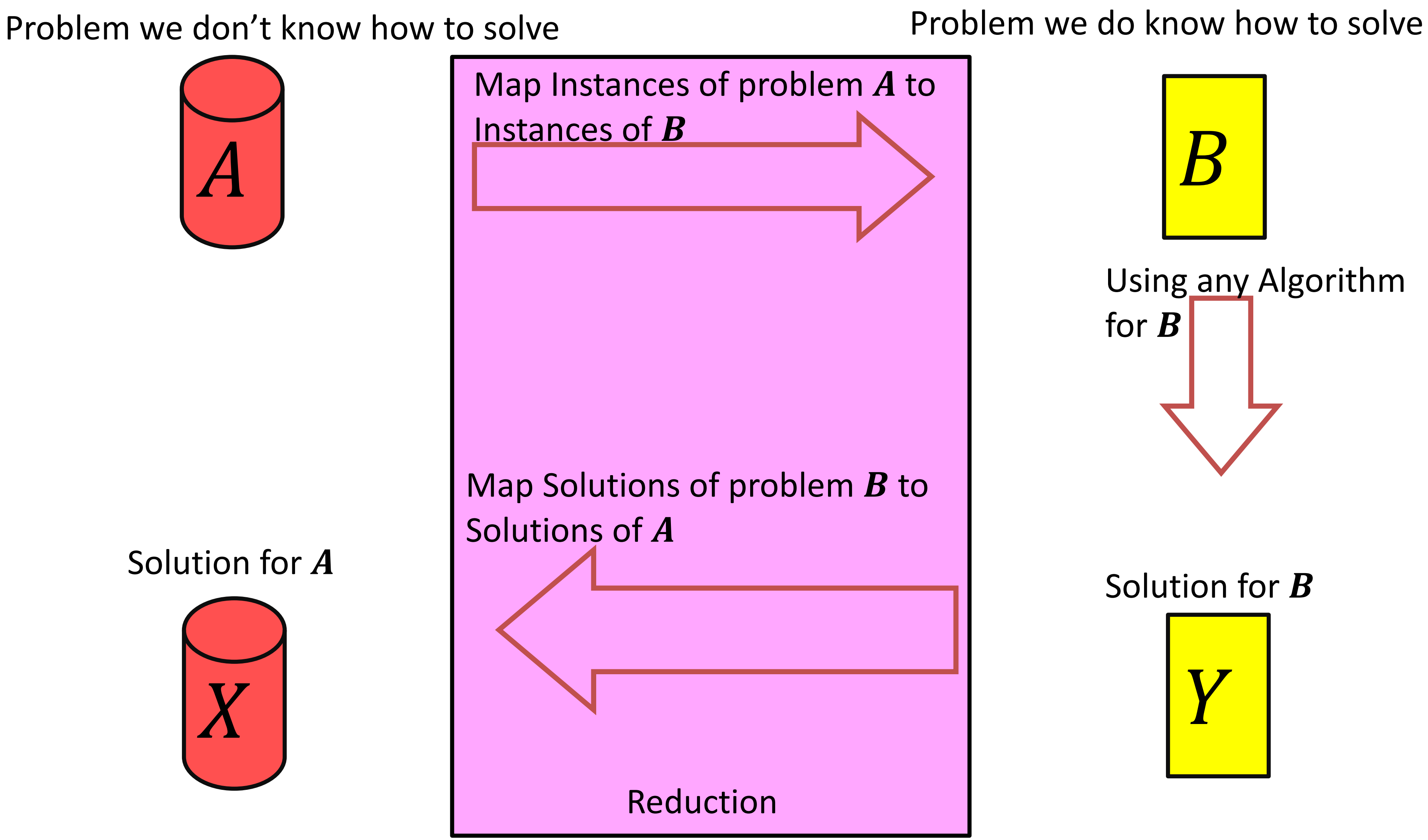
Homeworks, etc

- HW9 due tomorrow at 11pm
  - Written (use LaTeX)
  - Reductions
- Final Exam: Saturday, May 4, 2-5pm
  - Heavily from material since midterm
    - May ask for runtime of an algorithm, some knowledge of D&C
    - Won't directly ask you to solve recurrences
  - Practice final online by tomorrow
  - Review session: Wednesday, 4pm, MEC 205
- Office Hours: Tomorrow 11am-1pm, Wed 11am-12pm

Reductions

- Algorithm technique of supreme ultimate power
- Convert instance of problem A to an instance of Problem B
- Convert solution of problem B back to a solution of problem A

In General: Reduction



Worst-case lower-bound Proofs

Opening a door

A

Problem **A**

reduces to

B

Problem **B**

Lighting a fire

Alcohol, wood, matches

Y

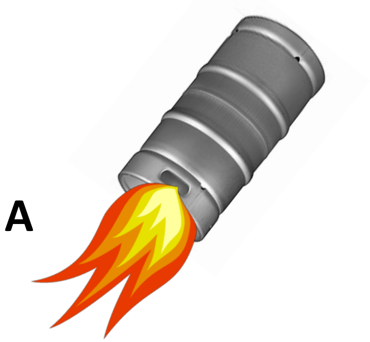
Algorithm for **B**

can be used to make

X

Algorithm for **A**

Keg cannon battering ram



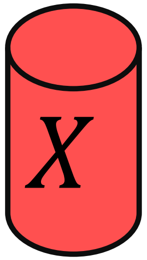
A is not a harder problem than B

$$A \leq B$$

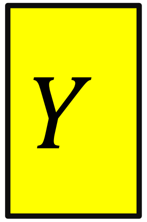
The name "reduces" is confusing: it is in the *opposite* direction of the making

Proof of Lower Bound by Reduction

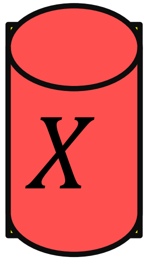
<u>To Show: Y is slow</u>



1. We know X is slow
(e.g., X = some way to open the door)



2. Assume Y is quick [toward contradiction]
(Y = some way to light a fire)

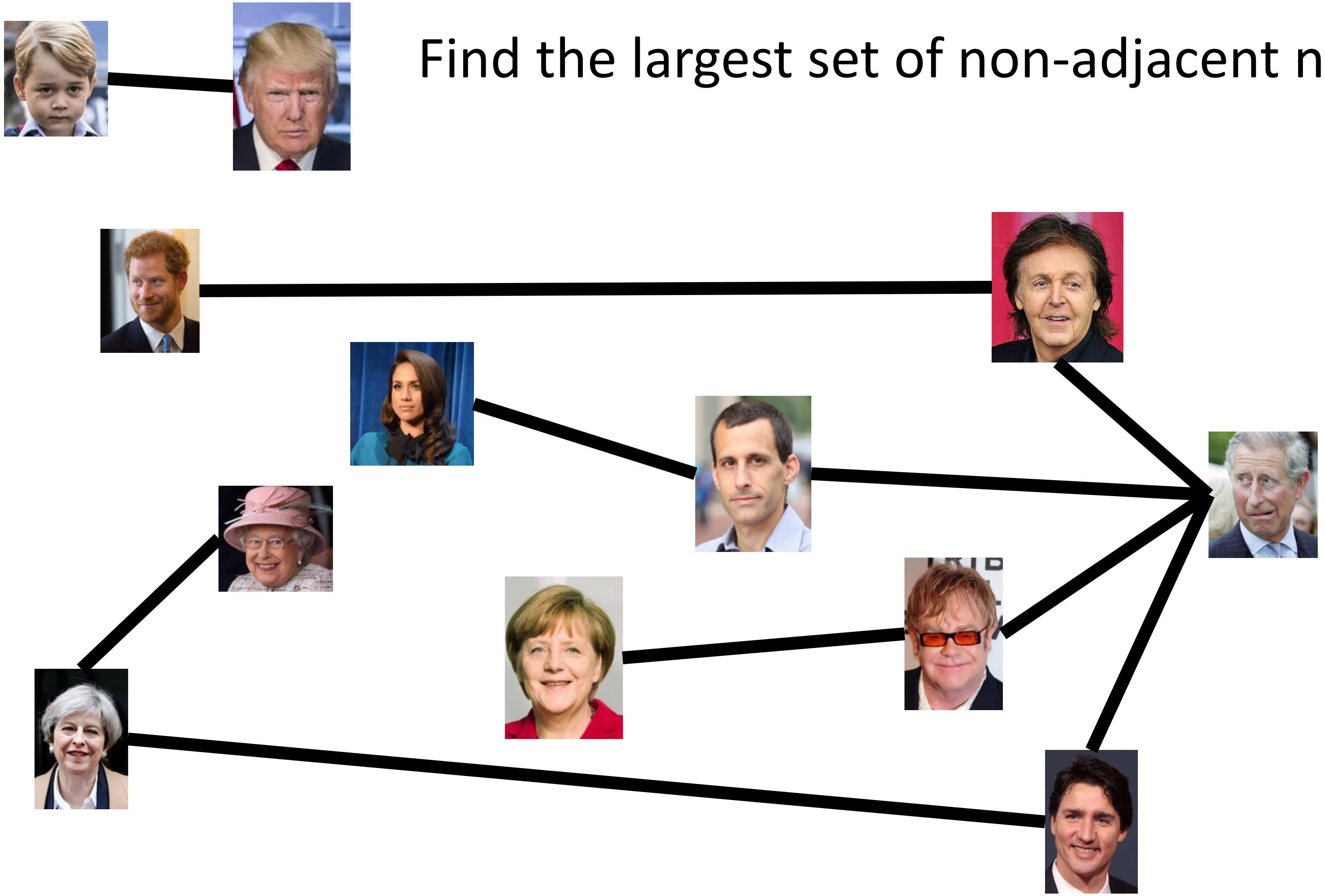


3. Show how to use Y to perform X quickly

4. X is slow, but Y could be used to perform X quickly
conclusion: Y must not actually be quick

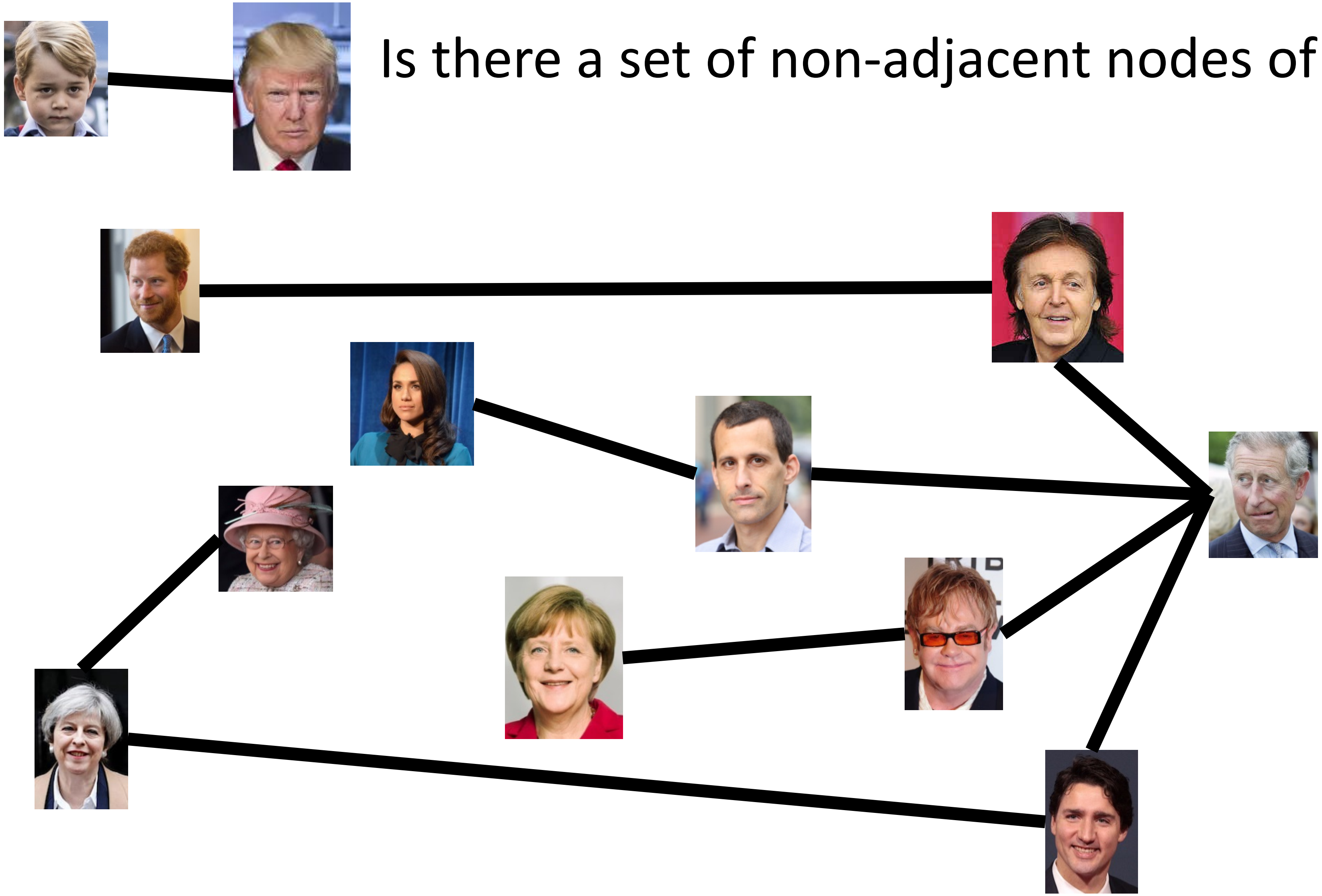
Max Independent Set

Find the largest set of non-adjacent nodes



k Independent Set

Is there a set of non-adjacent nodes of size k?



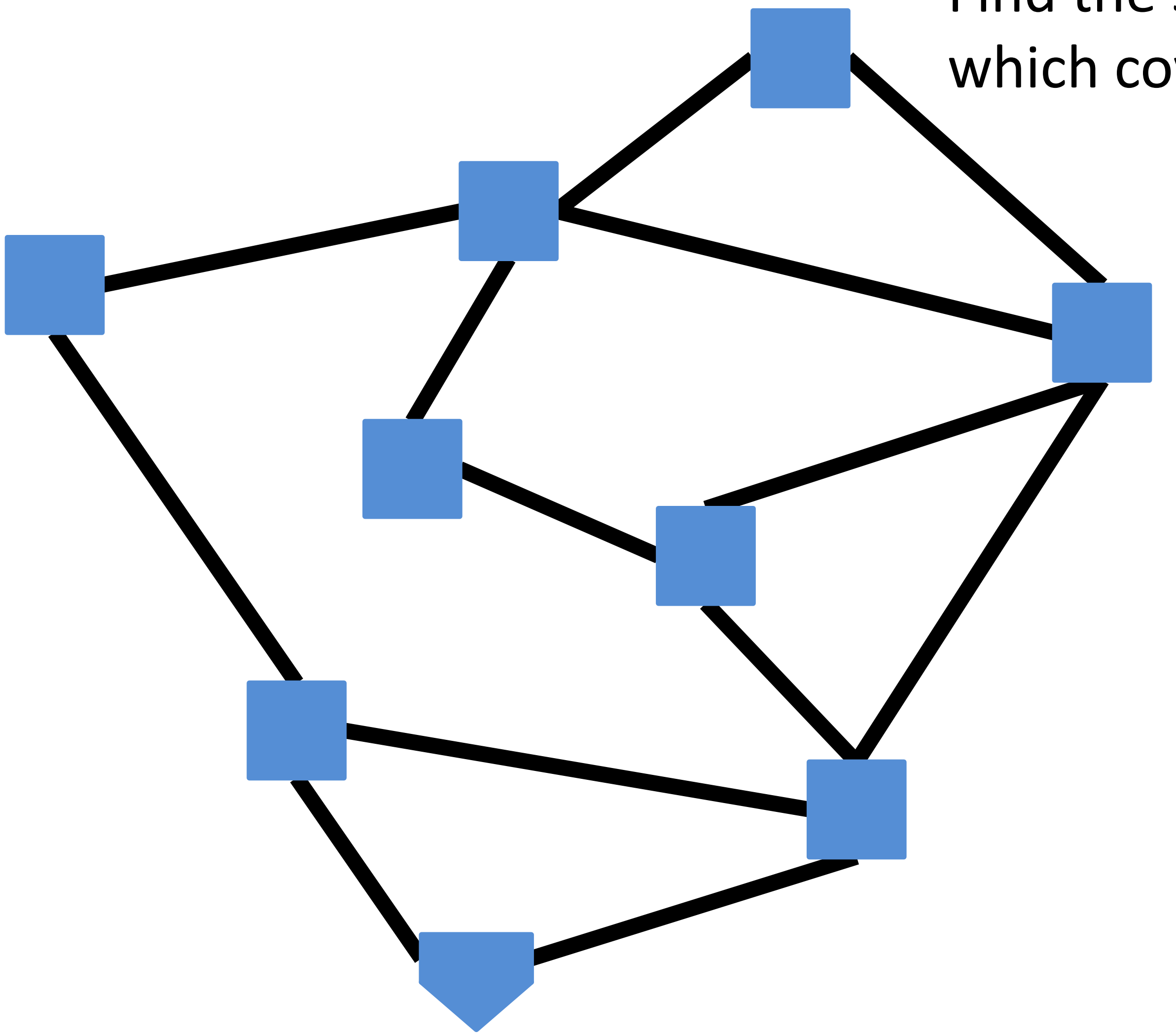
Maximum Independent Set

- Independent set: $S \subseteq V$ is an independent set if no two nodes in S share an edge
- Maximum Independent Set Problem: Given a graph $G = (V, E)$ find the maximum independent set S

k Independent Set

- Independent set: $S \subseteq V$ is an independent set if no two nodes in S share an edge
- k Independent Set Problem: Given a graph $G = (V, E)$ and a number k, **determine whether there is an independent set $\boldsymbol{S}$ of size $\boldsymbol{k}$**

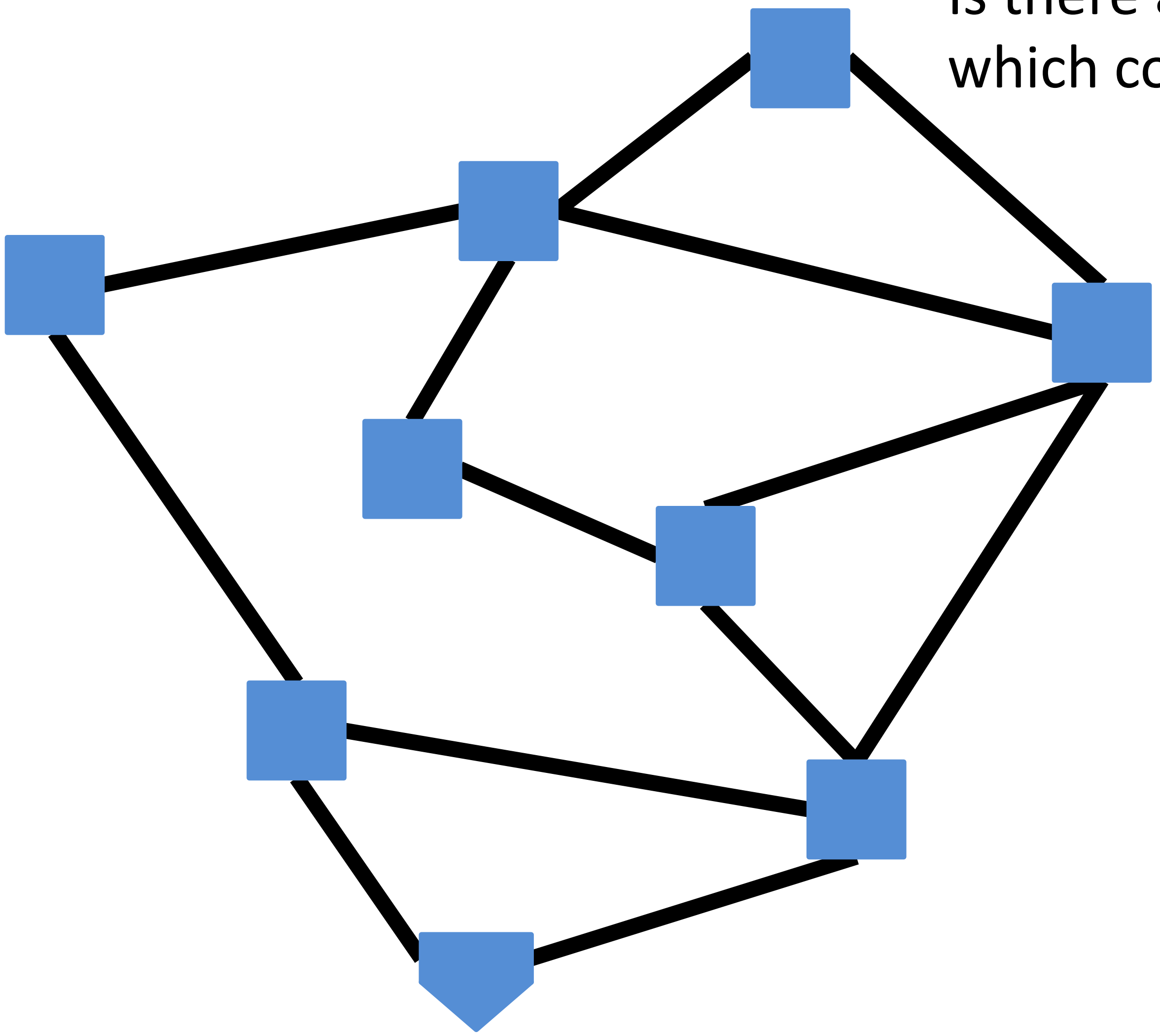
Min Vertex Cover



Find the smallest set of nodes which covers every edge

k Vertex Cover

Is there a set of nodes of size k which covers every edge?



Minimum Vertex Cover

- Vertex Cover: $C \subseteq V$ is a vertex cover if every edge in E has one of its endpoints in C
- Minimum Vertex Cover: Given a graph $G = (V, E)$ find the minimum vertex cover C

k Vertex Cover

- Vertex Cover: $C \subseteq V$ is a vertex cover if every edge in E has one of its endpoints in C
- k Vertex Cover: Given a graph $G = (V, E)$ and a number k, **determine whether there is a vertex cover $\boldsymbol{C}$ of size $\boldsymbol{k}$**

Problem Types

- Decision Problems:
  - Is there a solution?
    - Output is True/False
  - Is there a vertex cover of size k?
- Search Problems:
  - Find a solution
    - Output is complex
  - Give a vertex cover of size k
- Verification Problems:
  - Given a potential solution, is it valid?
    - Output is True/False
  - Is **this** a vertex cover of size k**?**

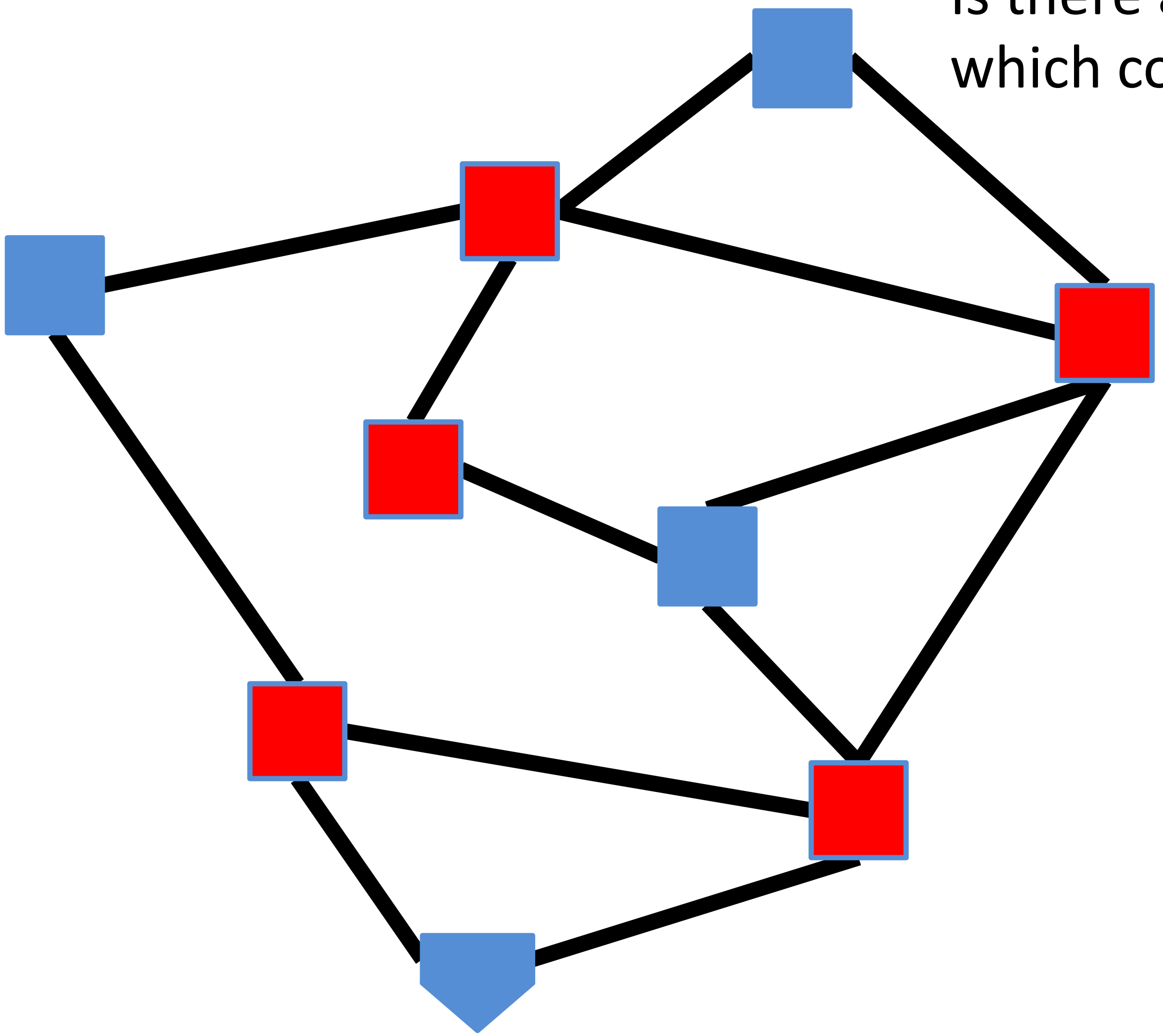
If we can solve this

Then we can solve this

Using a k-VertexCover decider to build a searcher

- Set $i = k - 1$
- Remove nodes (and incident edges) one at a time
- Check if there is a vertex cover of size i
  - If so, then that removed node was part of the k vertex cover, set $i = i - 1$
  - Else, it wasn't

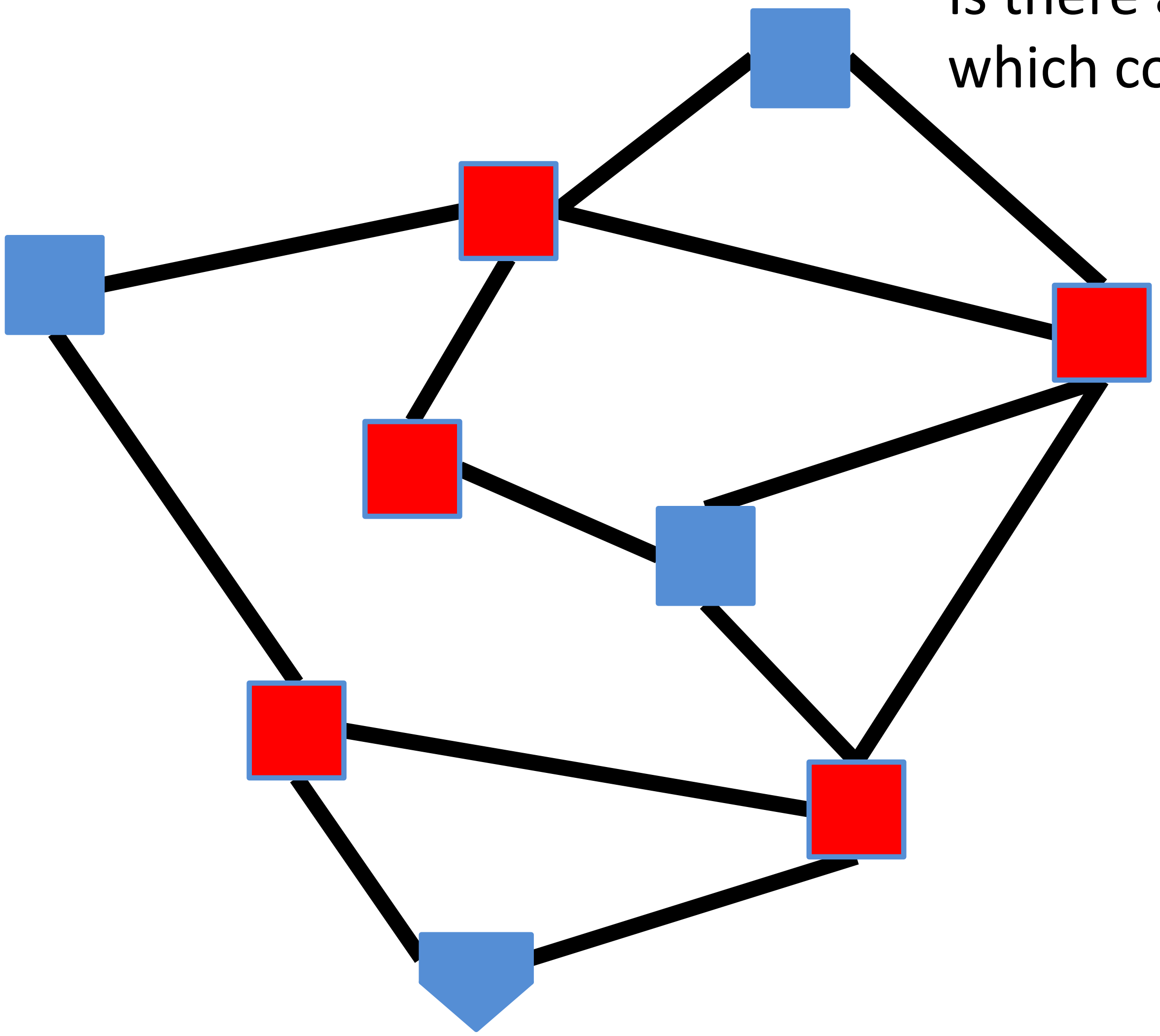
5 Vertex Cover (Decision)



Is there a set of nodes of size 5 which covers every edge?

Yes!

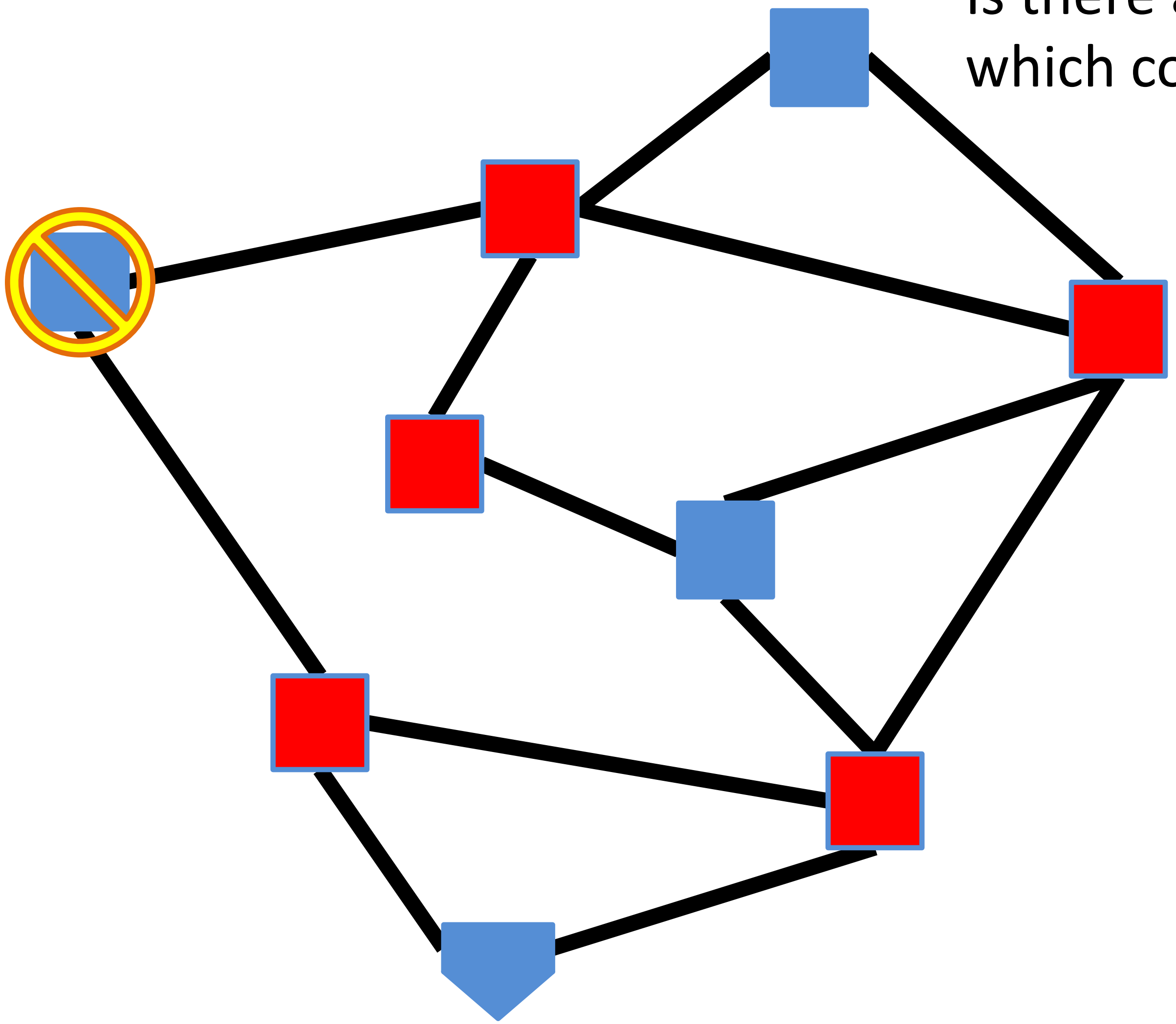
4 Vertex Cover (Decision)



Is there a set of nodes of size 4 which covers every edge?

No!

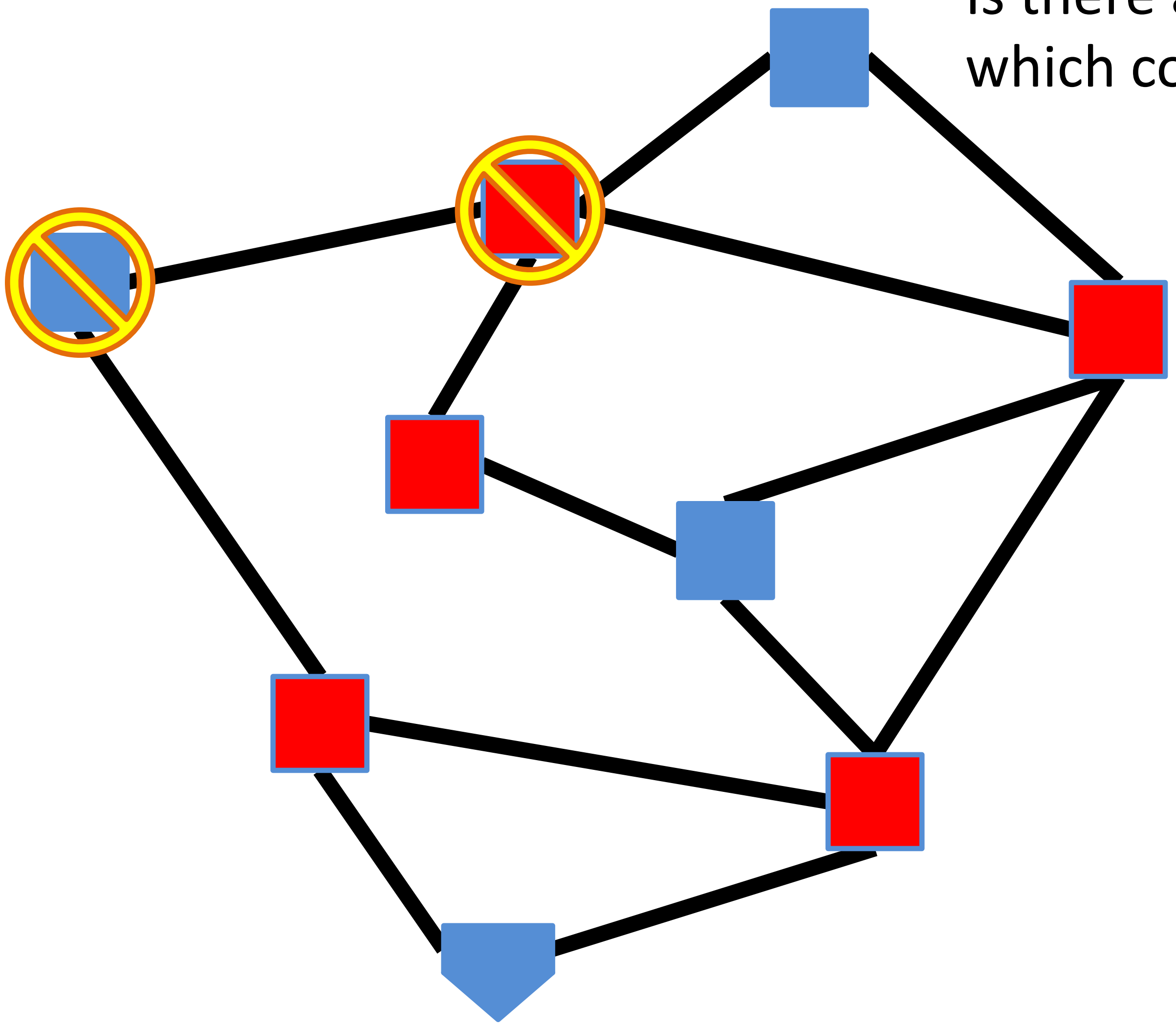
4 Vertex Cover (Decision)



Is there a set of nodes of size 4 which covers every edge?

No!

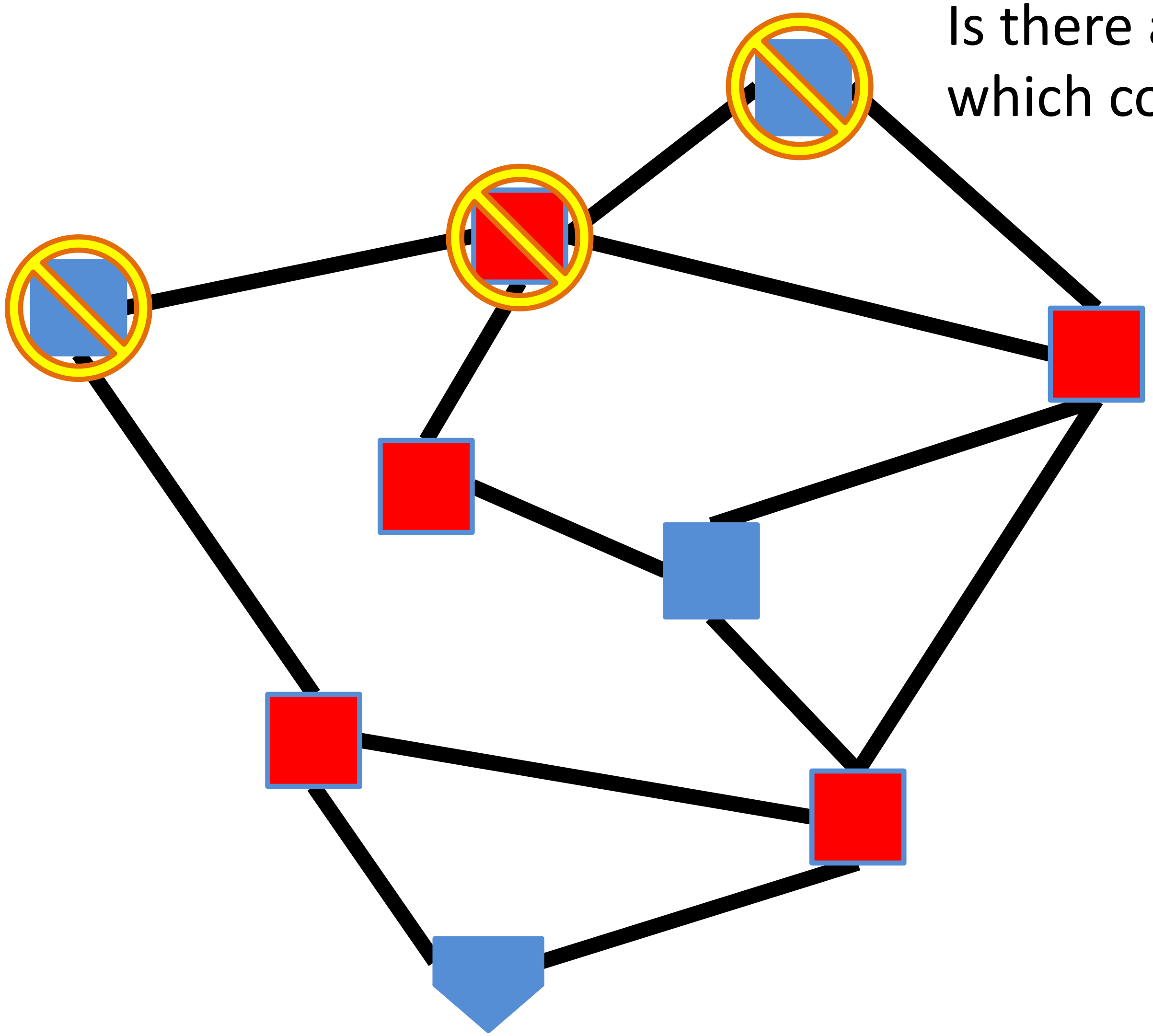
4 Vertex Cover (Decision)



Is there a set of nodes of size 4 which covers every edge?

Yes!

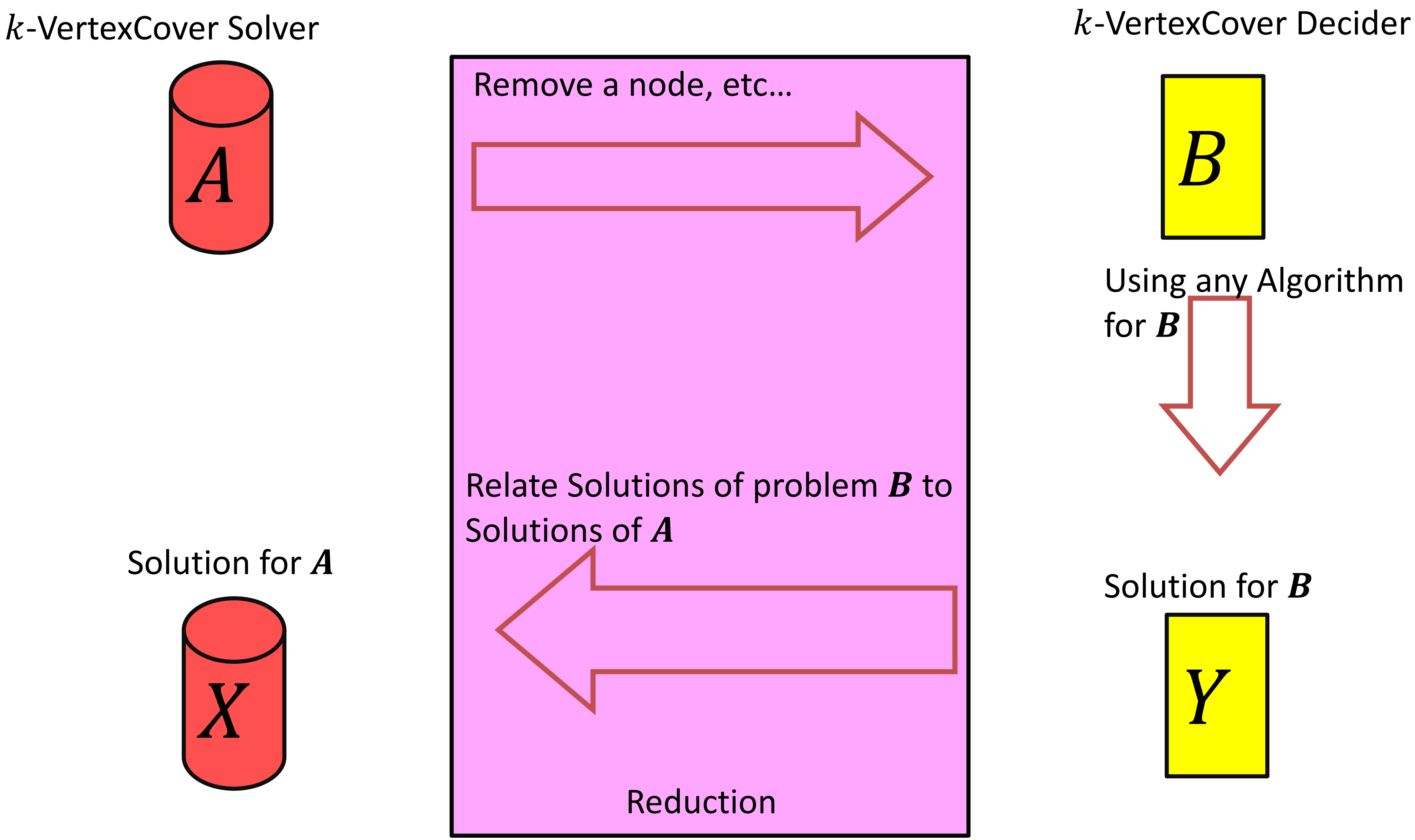
3 Vertex Cover (Decision)



Is there a set of nodes of size 3 which covers every edge?

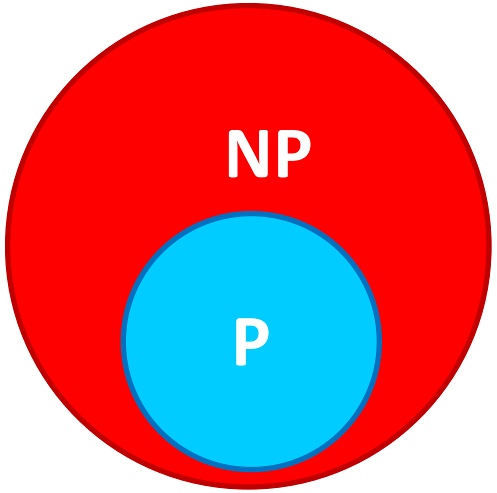
No!

Reduction


k-VertexCover Solver
A
Remove a node, etc…
k-VertexCover Decider
B
Using any Algorithm for $\boldsymbol{B}$
Relate Solutions of problem $\boldsymbol{B}$ to Solutions of $\boldsymbol{A}$
Solution for $\boldsymbol{A}$
X
Solution for $\boldsymbol{B}$
Y
Reduction

P vs NP

- P
  - Deterministic Polynomial Time
  - Problems solvable in polynomial time
    - $O(n^p)$ for some number p
- NP
  - Non-Deterministic Polynomial Time
  - Problems verifiable in polynomial time
    - $O(n^p)$ for some number p
- Open Problem: Does P=NP?
  - Certainly $P \subseteq NP$



k-Independent Set is NP

- To show: Given a potential solution, can we verify it in $O(n^p)$? $[n = V + E]$

How can we verify it?

1. Check that it's of size k $O(V)$
2. Check that it's an independent set $O(V^2)$

k-Vertex Cover is NP

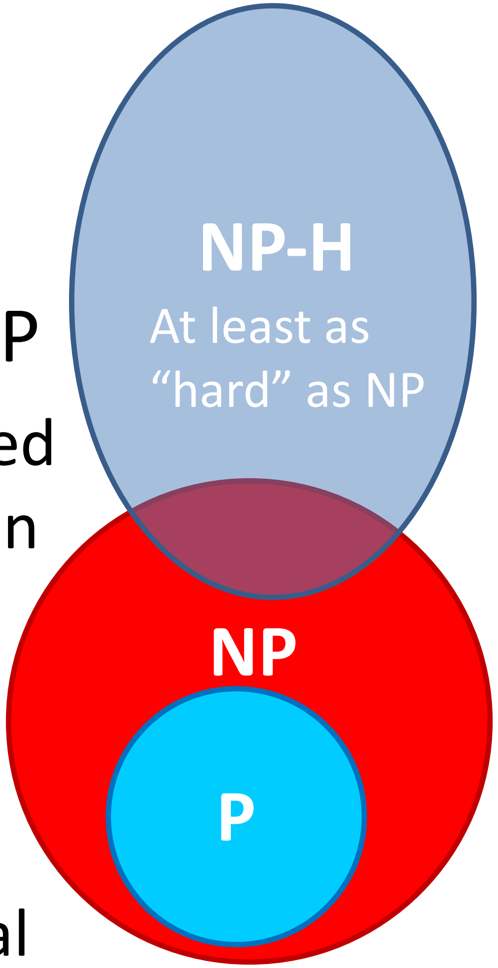
- To show: Given a potential solution, can we verify it in $O(n^p)$? $[n = V + E]$

How can we verify it?

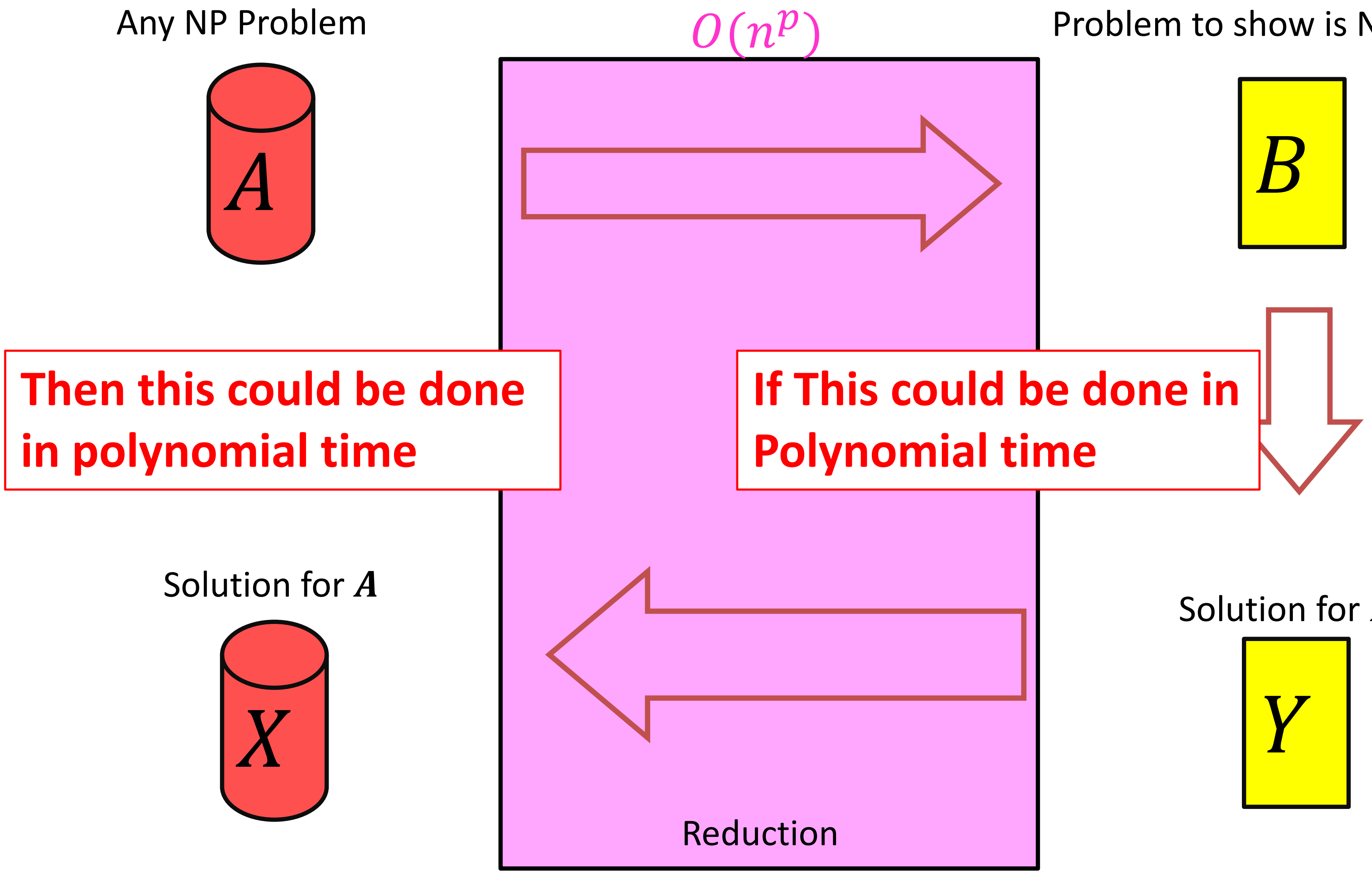
1. Check that it's of size k $O(V)$
2. Check that it's a Vertex Cover $O(E)$

NP-Hard

- How can we try to figure out if P=NP?
- Identify problems at least as “hard” as NP
  - If any of these “hard” problems can be solved in polynomial time, then all NP problems can be solved in polynomial time.
- Definition: NP-Hard:
  - B is NP-Hard if $\forall A \in NP, A \leq_p B$
  - $A \leq_p B$ means A reduces to B in polynomial time

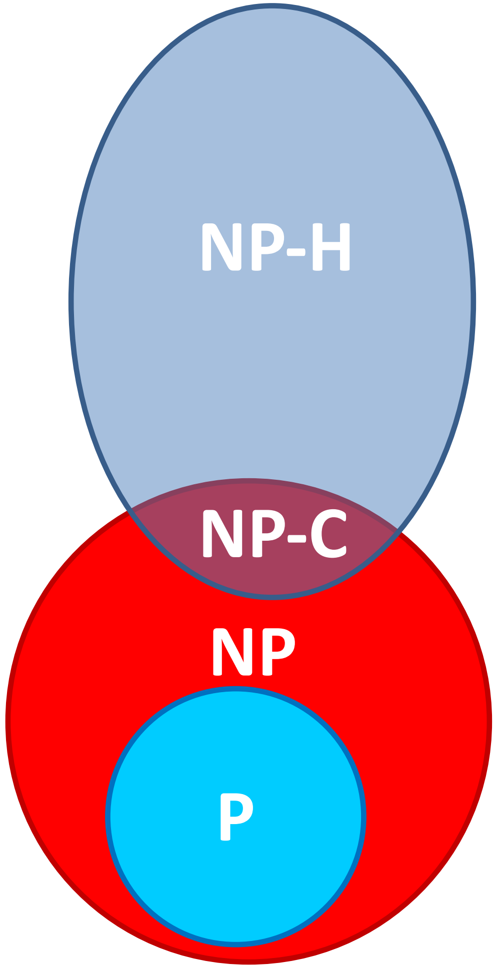


NP-Hardness Reduction

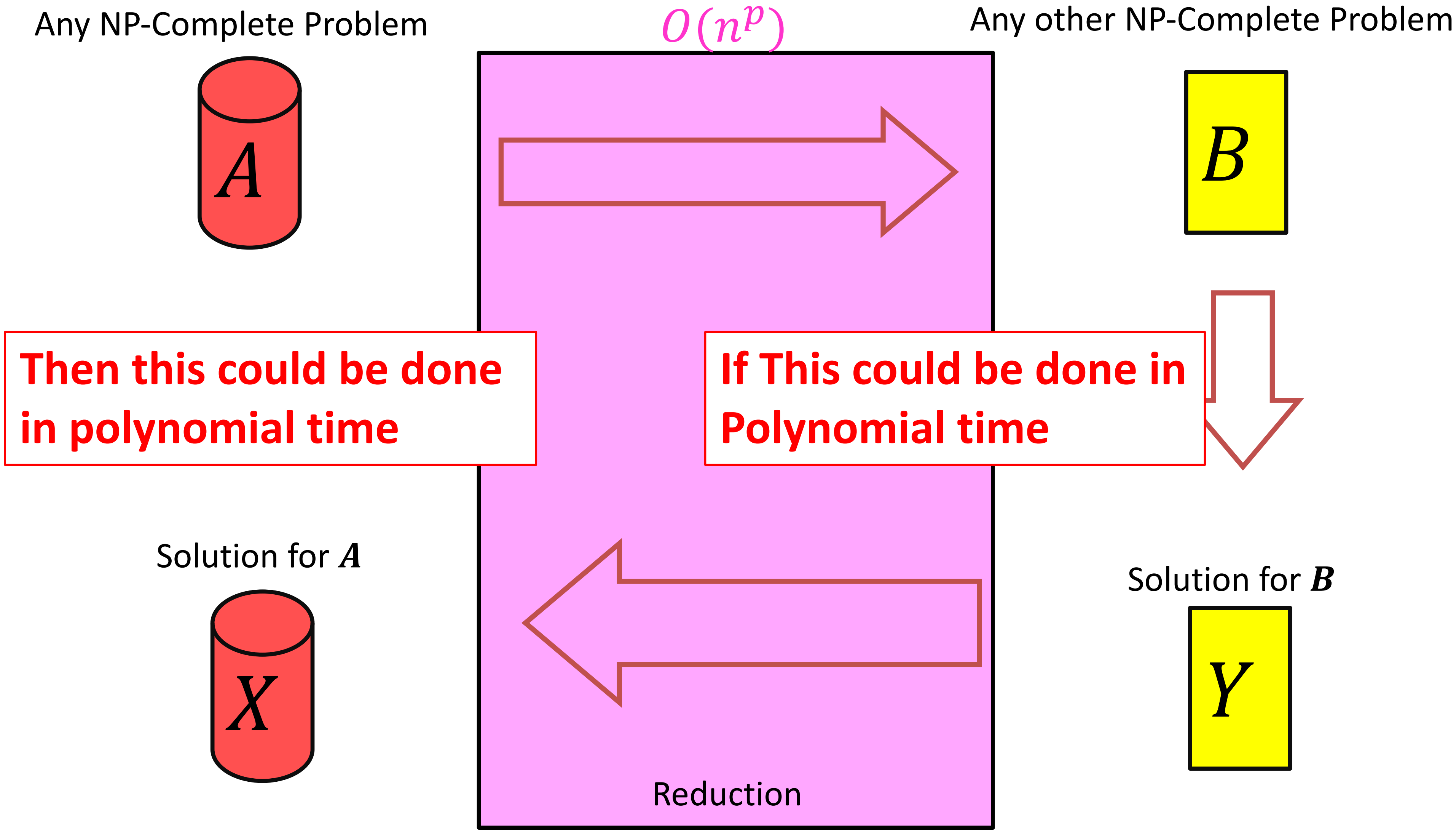


NP-Complete

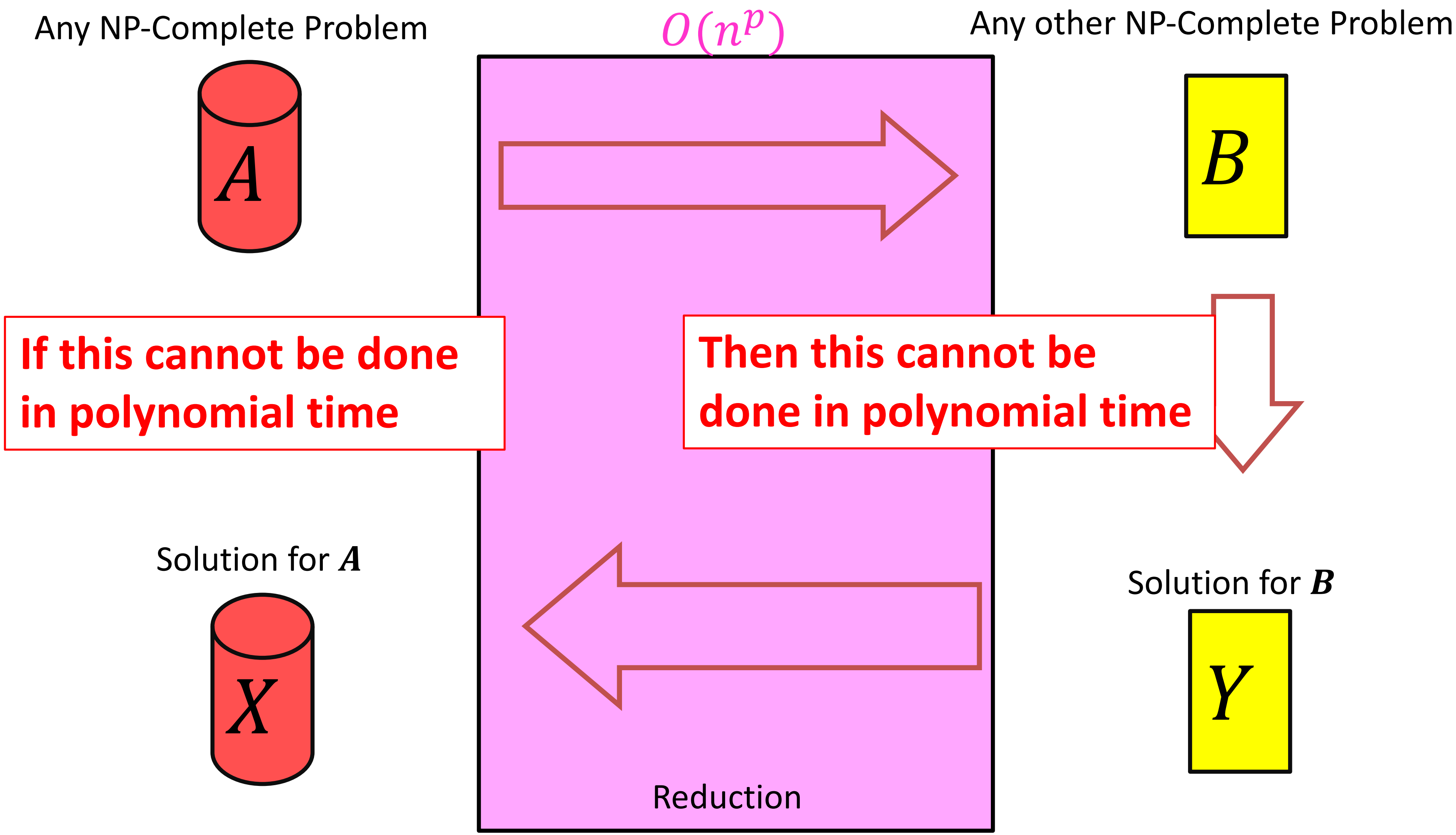
- “Together they stand, together they fall”
- Problems solvable in polynomial time iff ALL NP problems are
- NP-Complete = NP ∩ NP-Hard
- How to show a problem is NP-Complete?
  - Show it belongs to NP
    - Give a polynomial time verifier
  - Show it is NP-Hard
    - Give a reduction from another NP-H problem



NP-Completeness

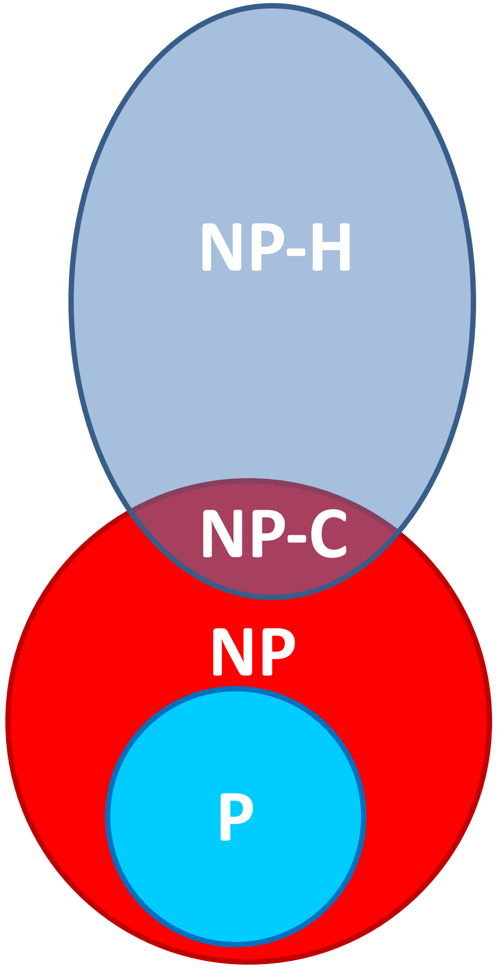


NP-Completeness



NP-Complete

- "Together they stand, together they fall"
- Problems solvable in polynomial time iff ALL NP problems are
- NP-Complete = NP ∩ NP-Hard
- How to show a problem is NP-Complete?
  - Show it belongs to NP
    - Give a polynomial time verifier
  - Show it is NP-Hard
    - Give a reduction from another NP-H problem



We now just need a FIRST NP-Hard problem

3-SAT

- Shown to be NP-Hard by Cook and Levin (independently)
- Given a 3-CNF formula (logical AND of clauses, each an OR of 3 variables), Is there an assignment of true/false to each variable to make the formula true?

$$\underbrace{(x \vee y \vee z)}_{\text{Clause}} \wedge (x \vee \bar{y} \vee y) \wedge (u \vee y \vee \bar{z}) \wedge (z \vee \bar{x} \vee u) \wedge (\bar{x} \vee \bar{y} \vee \bar{z})$$

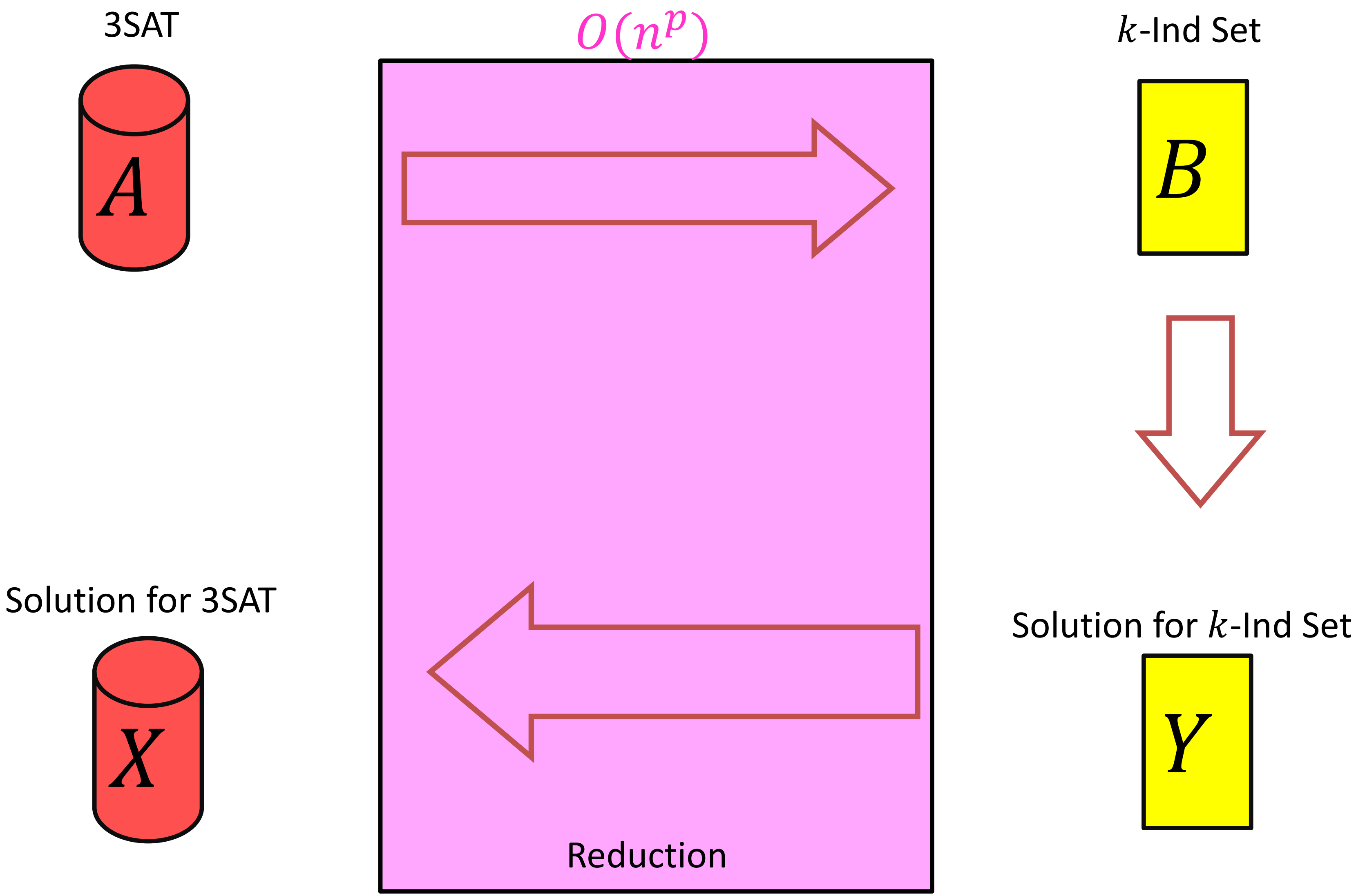
Variables

$$\boldsymbol{x = true}$$
$$\boldsymbol{y = false}$$
$$\boldsymbol{z = false}$$
$$\boldsymbol{u = true}$$

k-Independent Set is NP-Complete

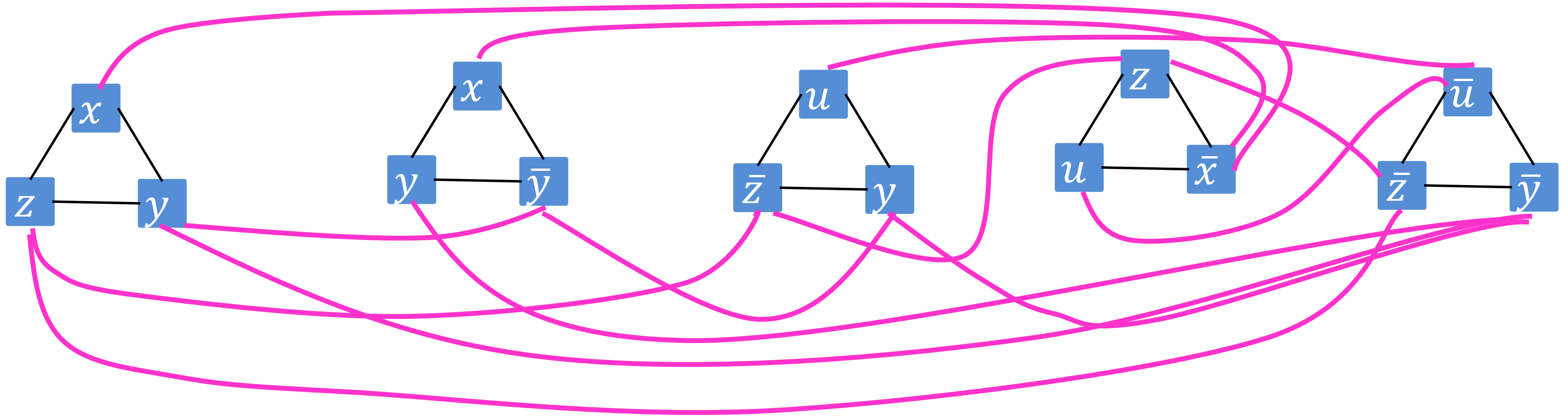
1. Show that it belongs to NP
   – Give a polynomial time verifier (see earlier slide)
2. Show it is NP-Hard
   – Give a reduction from a known NP-Hard problem
   – Show $3SAT \leq_p kIndSet$

$3SAT \leq_p kIndSet$



Instance of 3SAT to Instance of kIndSet

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



For each clause, produce a triangle graph with its three variables as nodes

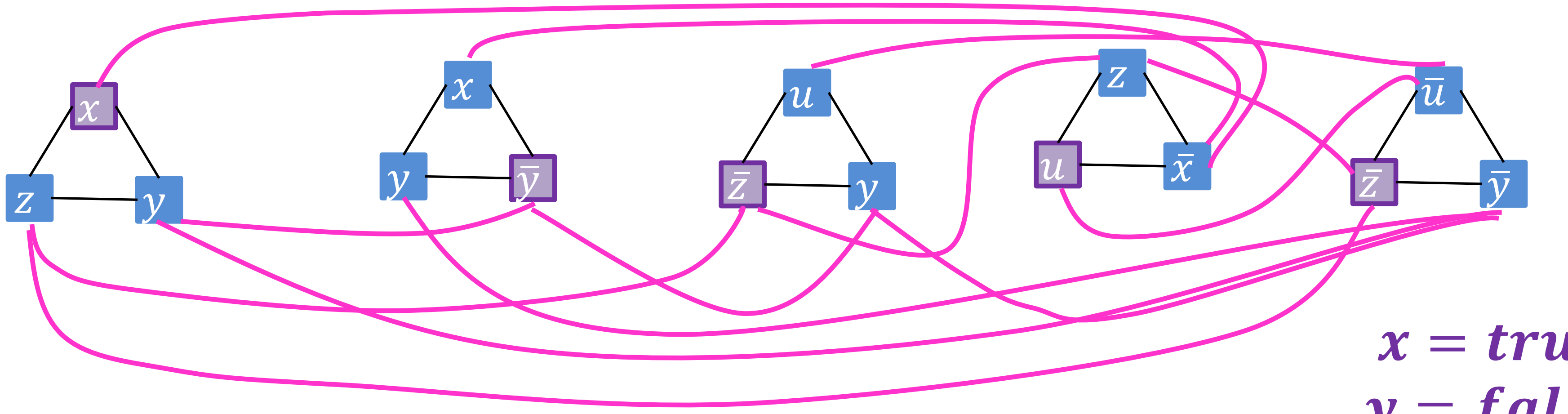
Connect each node to all of its opposites

Let k = number of clauses

There is a k-IndSet in this graph **iff** there is a satisfying assignment

kIndSet ⇒ Satisfying Assignment

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



$x = true$
$y = false$
$z = false$
$u = true$

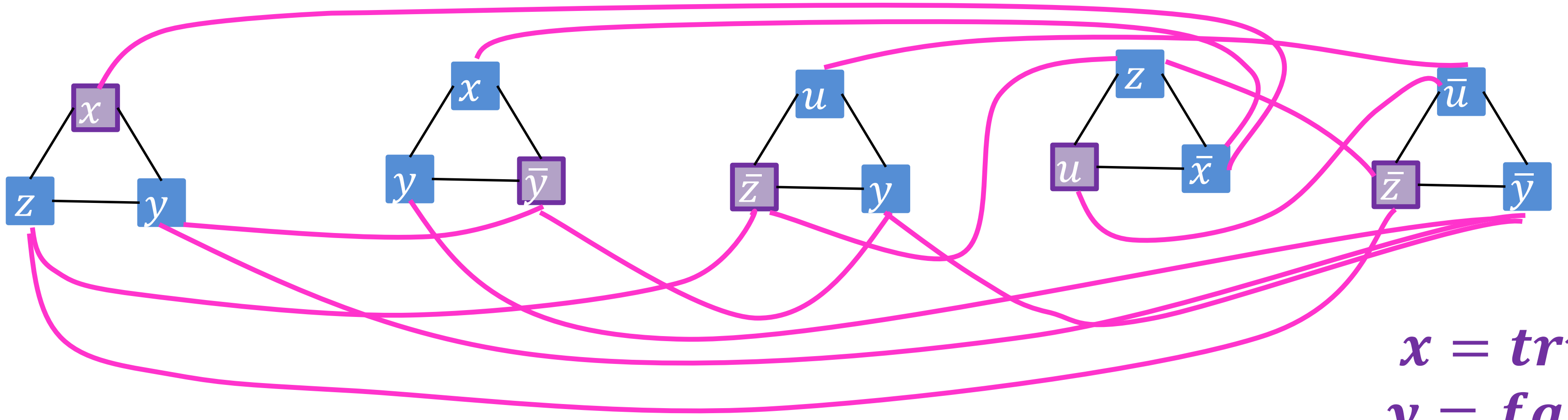
One node per triangle is in the Independent set:
because we can have exactly k total in the set,
and 2 in a triangle would be adjacent

If x is selected in some triangle, $\bar{x}$ is not selected in any triangle:
Because every x is adjacent to every $\bar{x}$

Set the variable which each included node represents to "true"

Satisfying Assignment ⇒ kIndSet

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



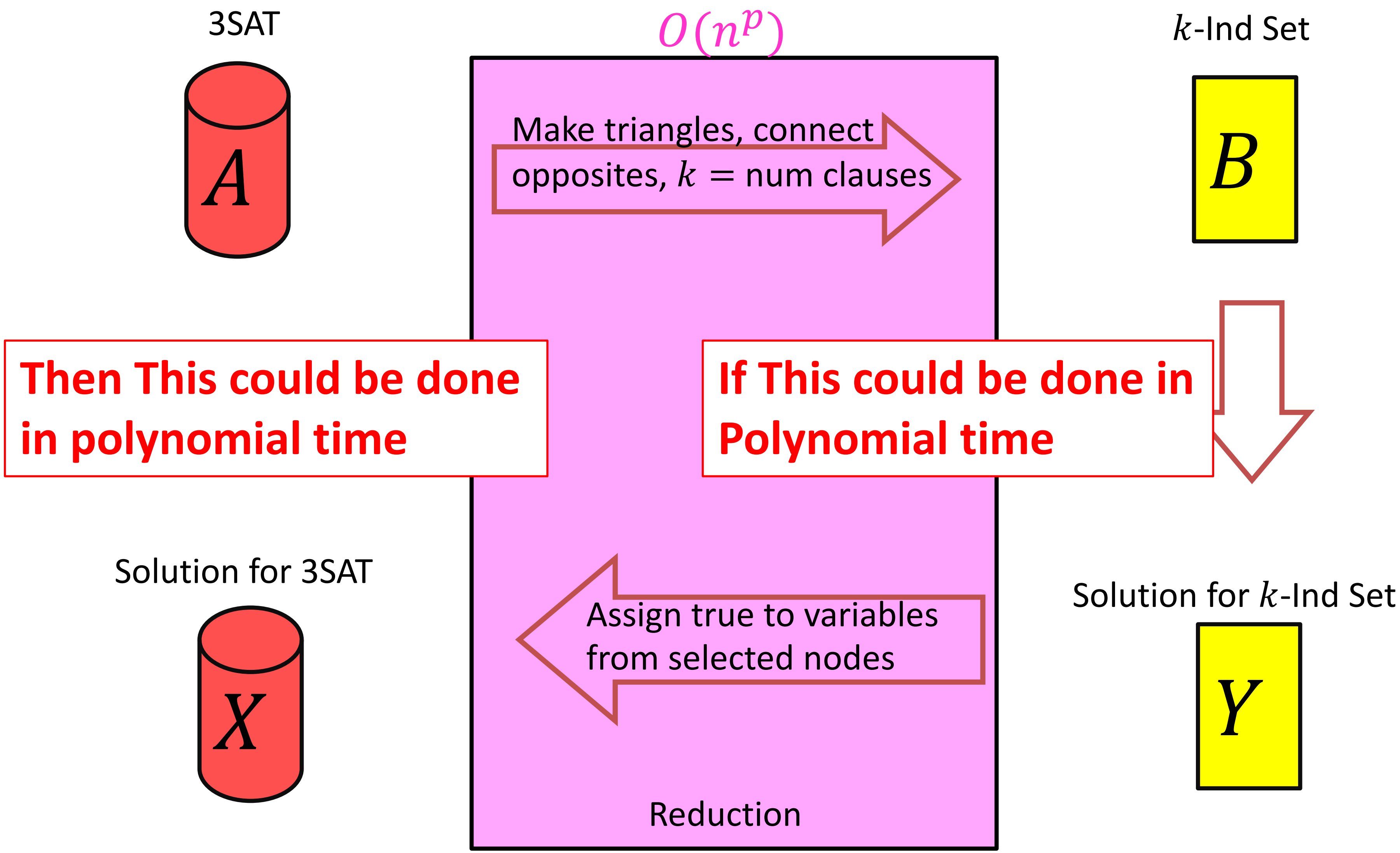
$\boldsymbol{x = true}$
$\boldsymbol{y = false}$
$\boldsymbol{z = false}$
$\boldsymbol{u = true}$

Use one true variable from the assignment for each triangle

The independent set has k nodes, because there are k clauses

If any variable x is true then $\bar{x}$ cannot be true

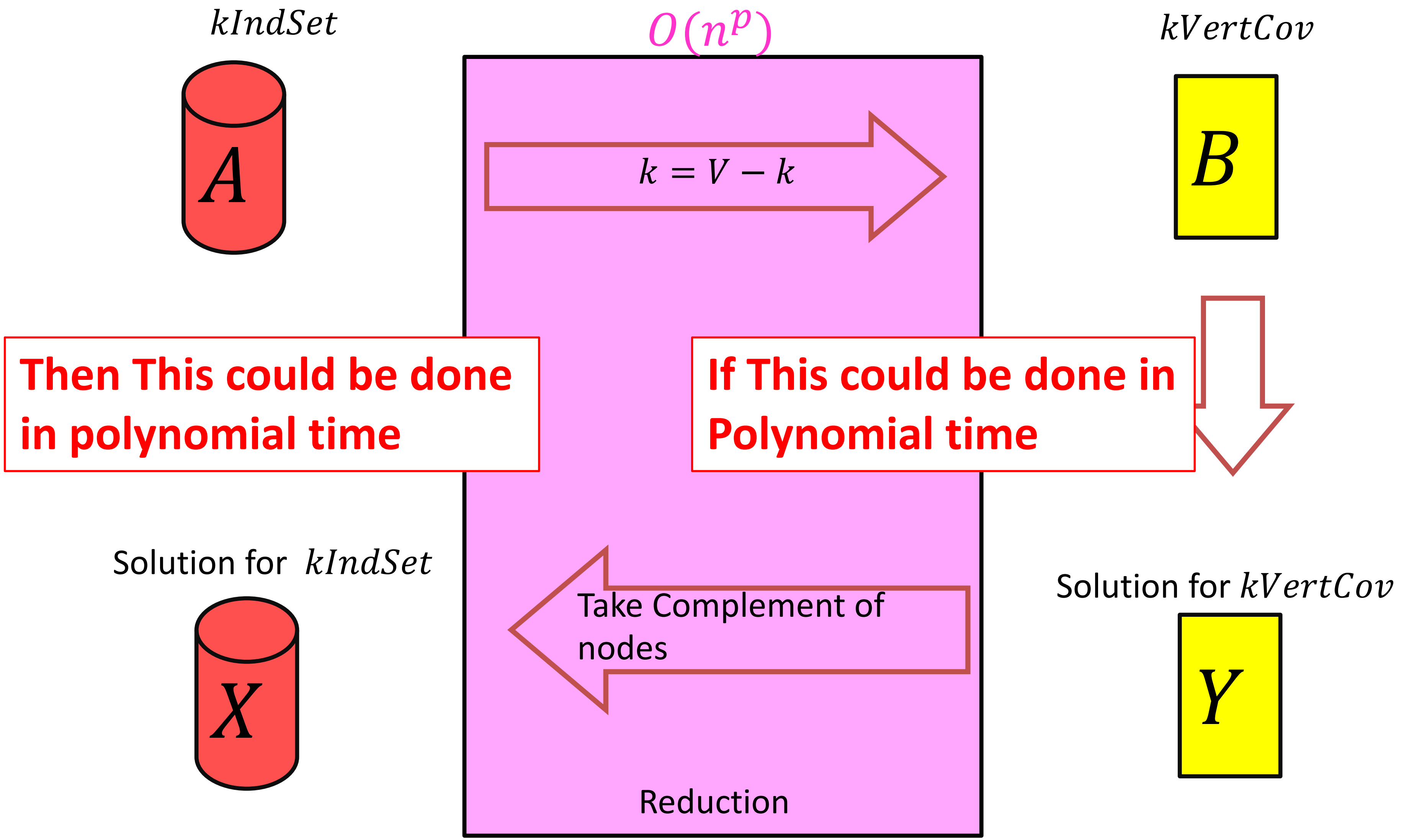
$3SAT \leq_p kIndSet$



k-Vertex Cover is NP-Complete

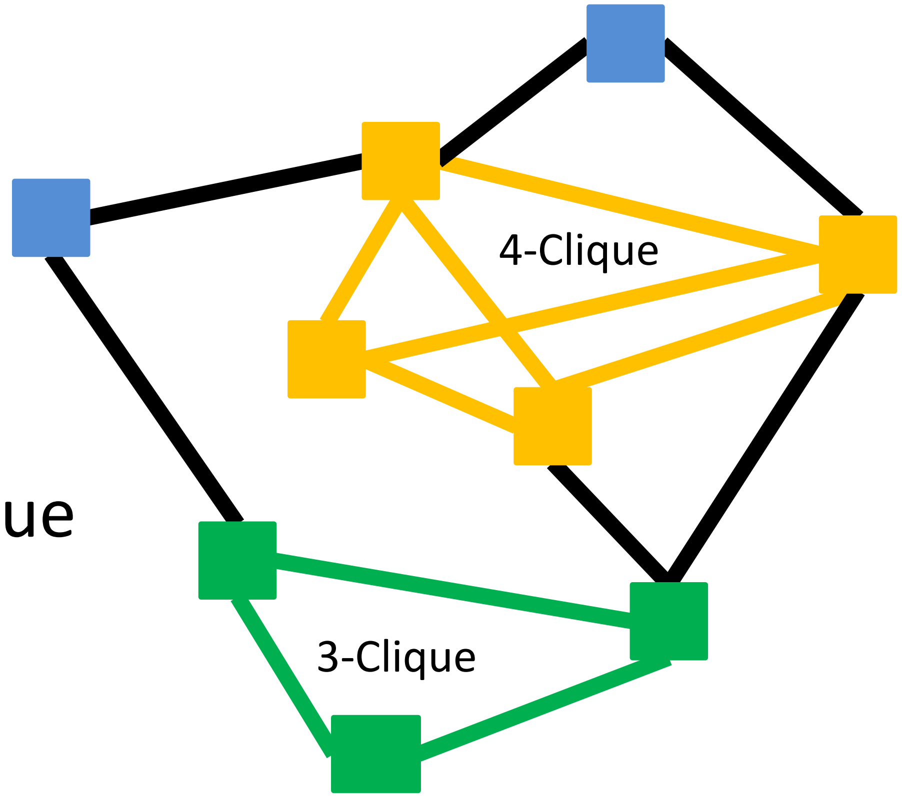
1. Show that it belongs to NP
   - Give a polynomial time verifier (see earlier slide)
2. Show it is NP-Hard
   - Give a reduction from a known NP-Hard problem
   - We showed $kIndSet \leq_p kVertCov$
     - (Last Class)

$kIndSet \leq_p kVertCov$



k-Clique Problem

- Clique: A complete subgraph
- k-Clique Problem:
  - Given a graph G and a number k, is there a clique of size k?

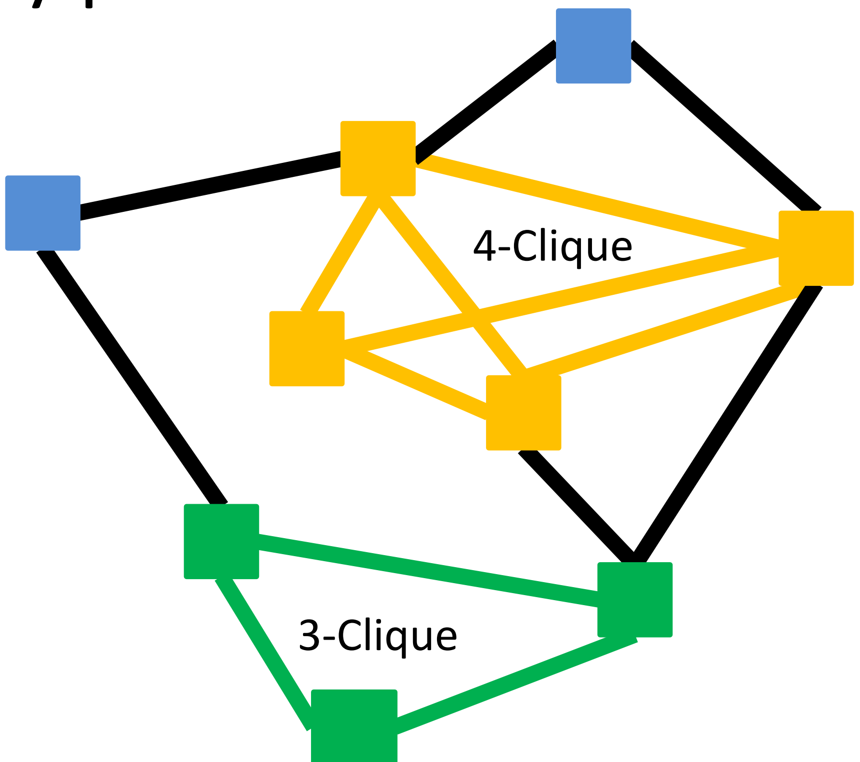


k-Clique is NP-Complete

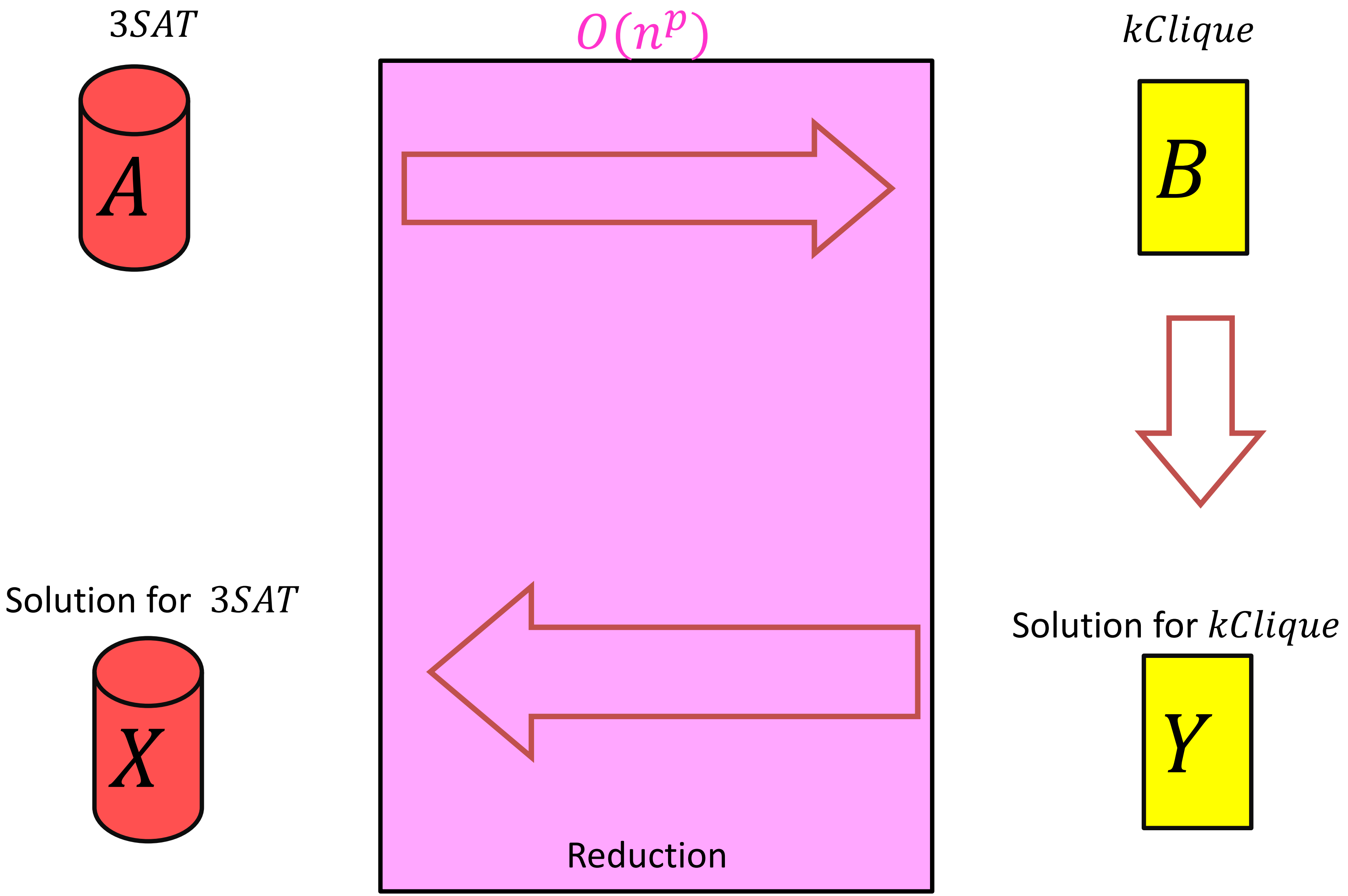
1. Show that it belongs to NP
   - Give a polynomial time verifier
2. Show it is NP-Hard
   - Give a reduction from a known NP-Hard problem
   - We will show $3SAT \leq_p kClique$

k-Clique is NP

1. Given a Graph and a potential solution
2. Check that the solution has k nodes
3. Check that every pair of nodes share an edge

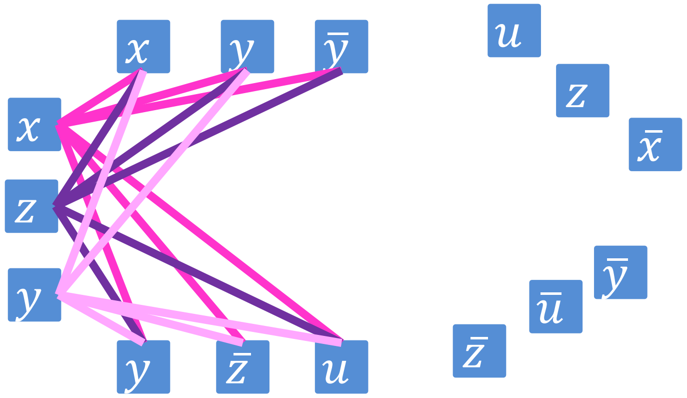


$3SAT \leq_p kClique$



Instance of 3SAT to Instance of kClique

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



(also do this for the other clauses, omitted due to clutter)

For each clause, produce a node for each of its three variables

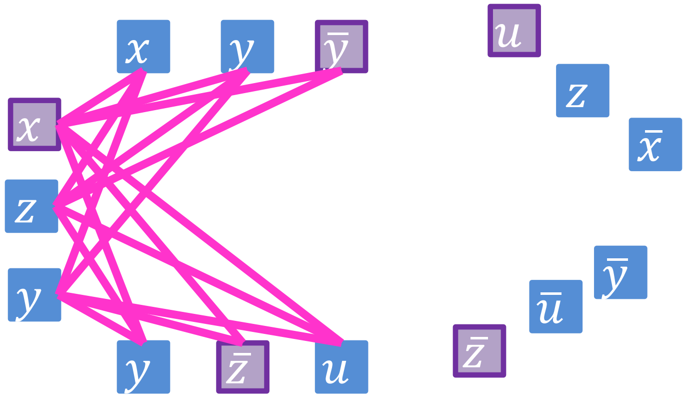
Connect each node to all non-contradictory nodes in the other clauses (i.e., anything that's not its negation)

Let k = number of clauses

There is a k-Clique in this graph **iff** there is a satisfying assignment

kClique $\Rightarrow$ Satisfying Assignment

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



$\boldsymbol{x = true}$
$\boldsymbol{y = false}$
$\boldsymbol{z = false}$
$\boldsymbol{u = true}$

There are k triplets in the graph, and no two nodes in the same triplet are adjacent

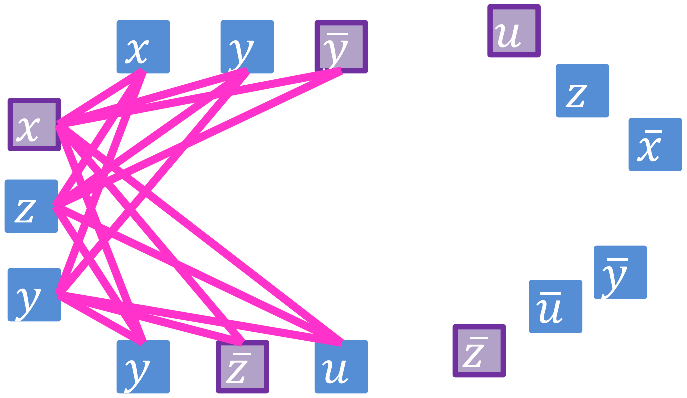
To have a k-Clique, must have one node from each triplet

Cannot select a node for both a variable and its negation

Therefore selection of nodes is a satisfying assignment

Satisfying Assignment$\Rightarrow k$Clique

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

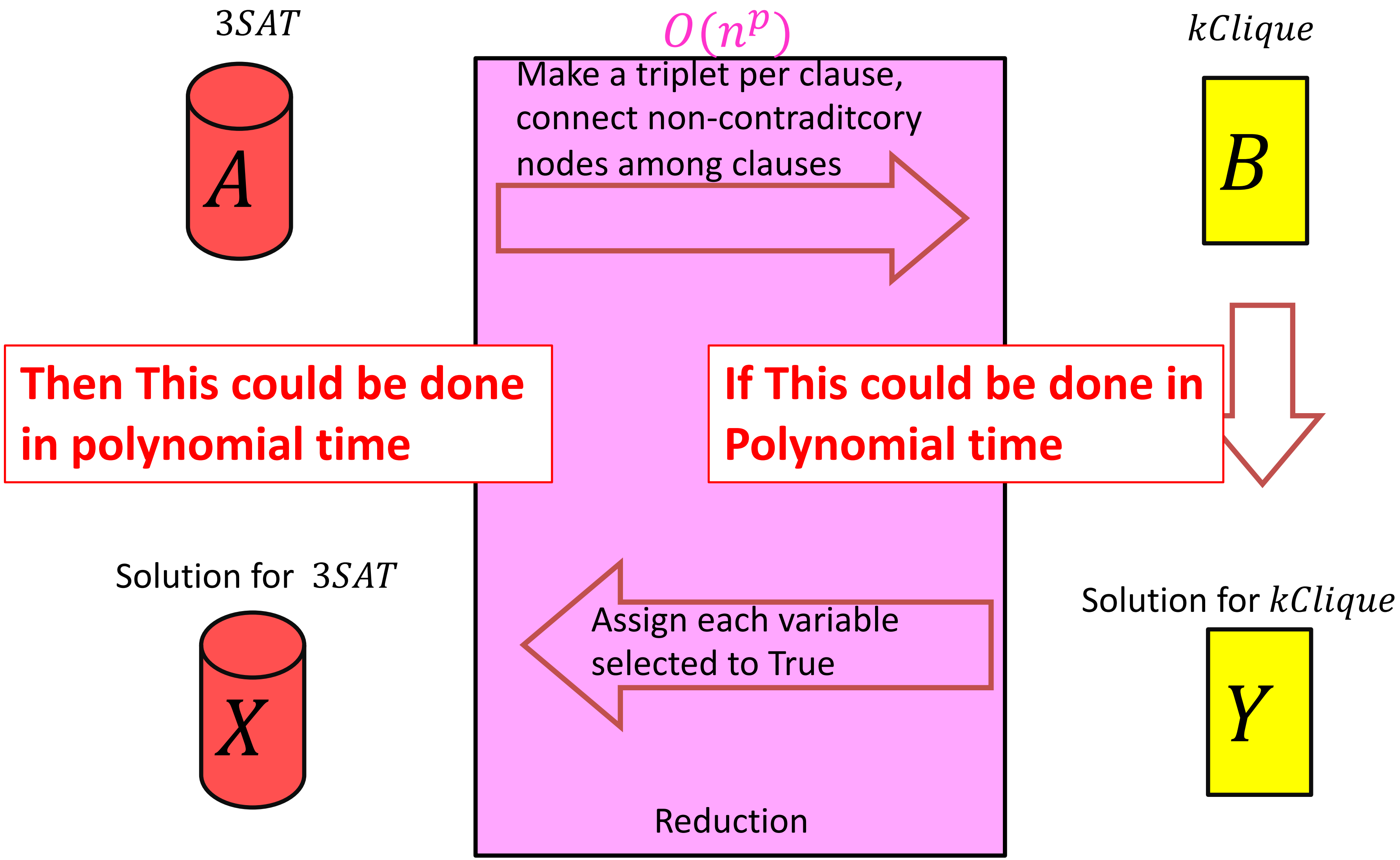


$\boldsymbol{x = true}$
$\boldsymbol{y = false}$
$\boldsymbol{z = false}$
$\boldsymbol{u = true}$

Select one node for a true variable from each clause

There will be k nodes selected
We can’t select both a node and its negation
All nodes will be non-contradictory, so they will be pairwise adjacent

$3SAT \leq_p kClique$



Academic Integrity

Differential Privacy

SECTIONS HOME SEARCH The New York Times


President Trump Expected to Shrink Bears Ears by as Much as 90 Percent


Ministers Look to Revive Martin Luther King's 1968 Poverty Campaign

Alabama's Disdain for Democrats Looms Over Its Senate Race


ABC Suspends Reporter Brian Ross Over Erroneous Report About Trump

As Computer Coding Classes Swell, So Does Cheating

ars TECHNICA BIZ & IT TECH SCIENCE POLICY CARS GAMING & CULTURE FORUMS SIGN IN

BIZ & IT —

Code copypasta increasingly common in CS education

Roughly 22 percent of Stanford honor code violations involve plagiarism in ...

RYAN PAUL - 2/12/2010, 5:11 PM

THE DAILY ILLINI

The independent student newspaper at the University of Illinois

NEWS SPORTS OPINIONS LIFE & CULTURE SPECIAL SECTIONS LONGFORM BUZZ CLASSIFIEDS

College of Engineering piloting program to combat cheating

Top Stories

Differential Privacy

- Gives a way to probabilistically answer questions about data without giving away its content
- You can get statistical certainty on the answer
- We're going to use a simple example

Scheme

- Flip a coin:
  - If Heads, respond "yes"
  - If Tails, truthfully answer an embarrassing question:
- Questions

- have you ever cheated?
- have ~~I~~ looked up a qn on S.O?
- do you like KPOP?
- anything illegal?

How does it work

- Assume everyone participates honestly
- We know 50% of “yes” answers were from the coin landing heads
  - If 100 people participate, eliminate 50 “yes” responses
  - Proportion of “yes” answers given by remaining “yes” answers over 50
- Consider a person who answers “no”
  - We know this person didn’t cheat
- Consider a person who answers “yes”
  - Most people who answered “yes” only did so because the coin landed heads
  - It’s still more likely that this person did not cheat

Example: How many people have streaked the lawn?

- Flip a coin:
  - If Heads, respond "yes"
  - If Tails, truthfully answer an embarrassing question:
    - Have you ever streaked the lawn?
      - On the slip of paper, put a 1 in ~~column 1~~ left side, put a 1 in ~~column 2~~ on right side if you answered yes (else a 0 ~~in column 2~~)
      - Pass the slip to your left

Does P=NP?

	$P \neq NP$	$P = NP$	Ind	DC	DK	DK and DC	other
2002	61(61%)	9(9%)	4(4%)	1(1%)	22(22%)	0(0%)	3(3%)
2012	126 (83%)	12 (9%)	5 (3%)	5 (3%)	1(0.6%)	1 (0.6%)	1 (0.6%)

When Will P=NP be resolved?

	02–09	10–19	20–29	30–39	40–49	50–59	60–69	70–79
2002	5(5%)	12(12%)	13(13%)	10(10%)	5(5%)	12 (12%)	4(4%)	0(0%)
2012	0(0%)	2(.01%)	17(11%)	18(12%)	5(3%)	10 (6.5%)	10 (6.5%)	9(6%)

	80–89	90–99	100–109	110–119	150–159	2200–3000	4000–4100
2002	1(1%)	0(0%)	0(0%)	0(0%)	0(0%)	5(5%)	0(0%)
2012	4(3%)	5(3%)	2(1.2%)	5(3%)	2(1.2%)	3(2%)	3(2%)

	Long Time	Never	Don't Know	Sooner than 2100	Later than 2100
2002	0(0%)	5(5%)	21(21%)	62(62%)	17 (17%)
2012	22(14%)	5(3%)	8(5%)	81(53%)	63 (41%)

Notable Statements on P vs NP

Scott Aaronson I believe $\mathrm{P} \neq \mathrm{NP}$ on basically the same grounds that I think I won't be devoured tomorrow by a 500-foot-tall robotic marmoset from Venus, despite my lack of proof in both cases.

Suggested rephrased question:

will humans manage to prove $\mathrm{P} \neq \mathrm{NP}$ *before they either kill themselves out or are transcended by superintelligent cyborgs? And if the latter, will the cyborgs be able to prove* $\mathrm{P} \neq \mathrm{NP}$*?*

Neil Immerman $\mathrm{P} \neq \mathrm{NP}$ will be resolved somewhere between 2017 and 2034, using some combination of logic, algebra, and combinatorics.

Donald Knuth: (Retired from Stanford) It will be solved by either 2048 or 4096. I am currently somewhat pessimistic. The outcome will be the truly worst case scenario: namely that someone will prove "P=NP because there are only finitely many obstructions to the opposite hypothesis"; hence there will exists a polynomial time solution to SAT but we will never know its complexity!