Lecture 33: More Reductions, Hard Decisions

COSC 311 *Algorithms*, Fall 2022

Announcement

Job Candidate Talk **TODAY**

Victoria Dean, Carnegie Mellon University

> *Bridging Reinforcement Learning and Robotics: Efficient Training and Shared Evaluation*

- 4:00 in SCCE A131
- Refreshments at 3:30 in SCCE C209

Remaining Coursework

1. Lecture Ticket for Monday (posted today)
2. Homework 6 due Next Friday (posted this weekend)
3. Final Exam: Friday Dec. 16 9:00
    - official announcement next week

Today

1. Independent Sets and Vertex Covers
2. Decision Problems and Reductions
3. Hard Decision Problems?

$O(N)$, $O(N^2)$, $O(N^{100})$

Last Time

"efficient" $\Omega(2^N)$ ✗ not poly.

1. Coarse notion of efficiency:
   - an algorithm is **polynomial time** if its worst-case running time is $O(N^c)$ for some constant c, N = input size

Max. Bip. Match

2. Notion of **reduction**
   - transform instances of problem A to instances of problem B Max Flow
   - solution to B reveals solution to A

Application. Maximum Bipartite Matching

- solved via reduction to Maximum Flow

Today

1. Independent Sets & Vertex Covers
2. Decision Problems
3. Easy and Hard Decision Problems

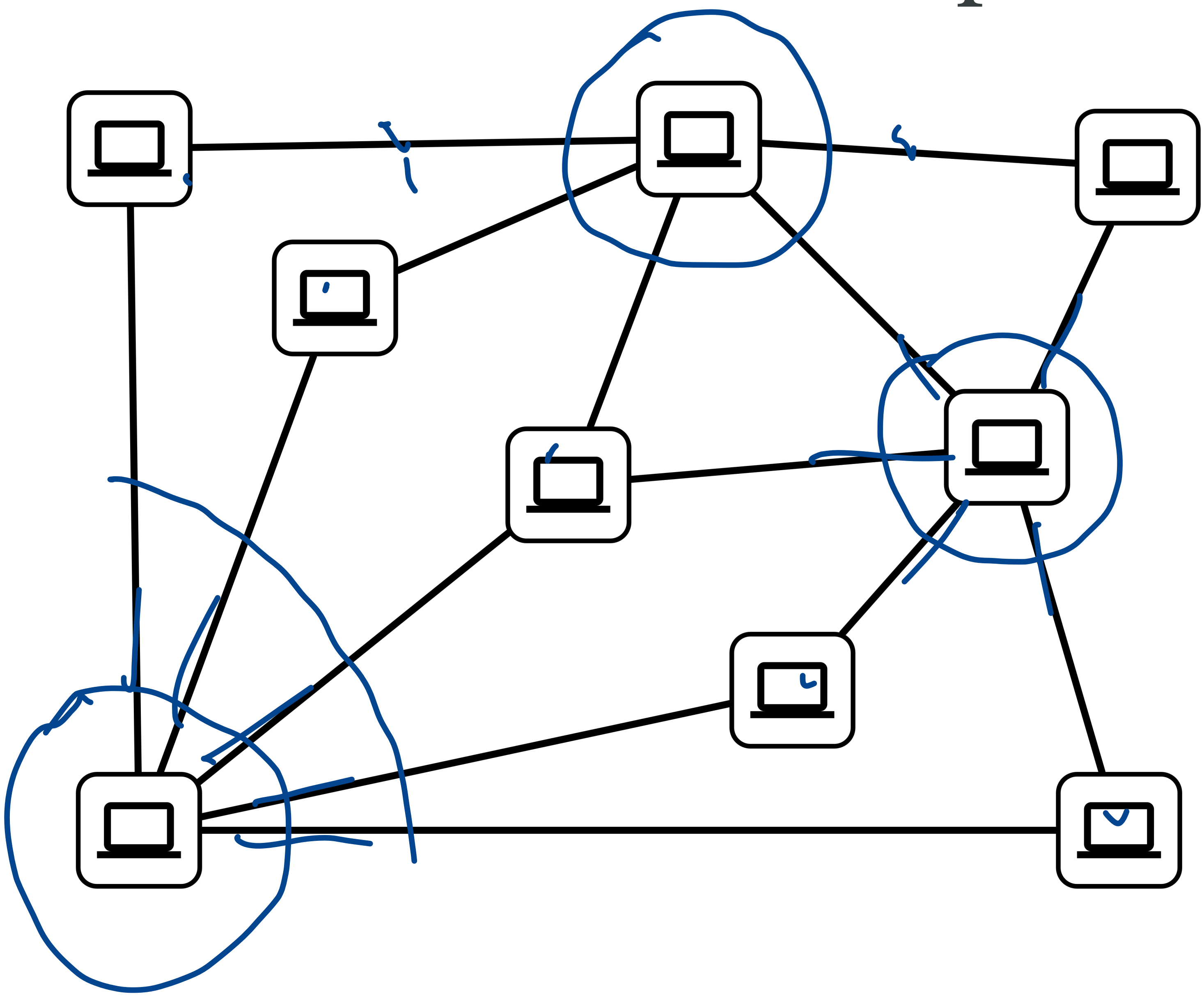
Virus Protection

Task: Install virus protection software on computers in a network

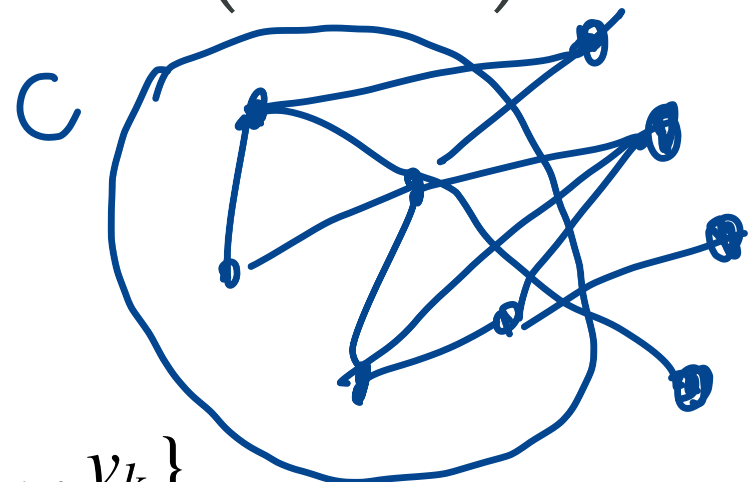
- if two computers are connected, at least one endpoint must be protected
- software licenses are expensive!

Goal: Install software on as few computers as possible while protecting the network

Virus Protection Example



Minimum Vertex Cover (MVC)



Input:

- Graph $G = (V, E)$

Output:

- A **vertex cover** $C = \{v_1, v_2, \ldots, v_k\}$
  - every edge $e = (u, v) \in E$ has $u \in C$ or $v \in C$
- C is a vertex cover of minimal size
  - there is no vertex cover of size $\ell \leq k$

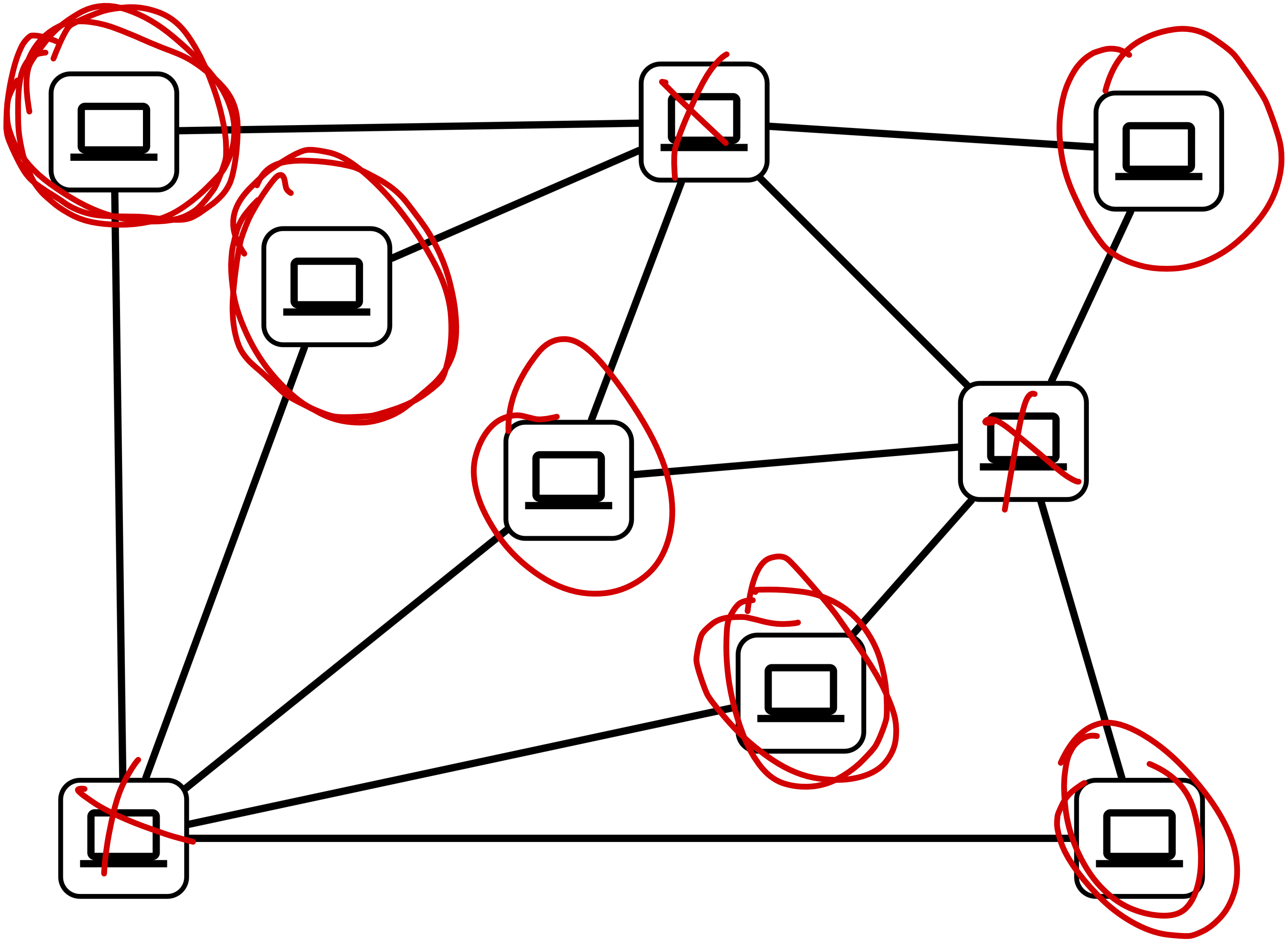
Firmware Upgrade

Task: Update firmware on computers in a network

- if two adjacent computers get updated, network connection between them must be manually reset
- resetting a network connection is annoying!

Goal: Update firmware on as many computers as possible without having to reset any network connections

Firmware Upgrade Example

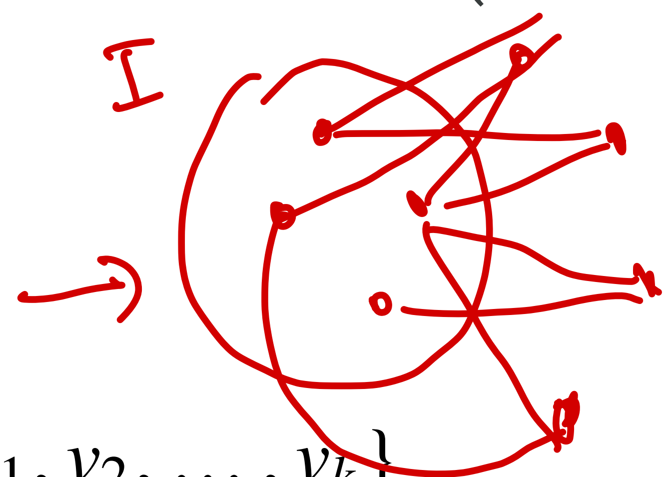


Maximum Independent Set (MaxIS)

Input:

- Graph $G = (V, E)$

no edges in here



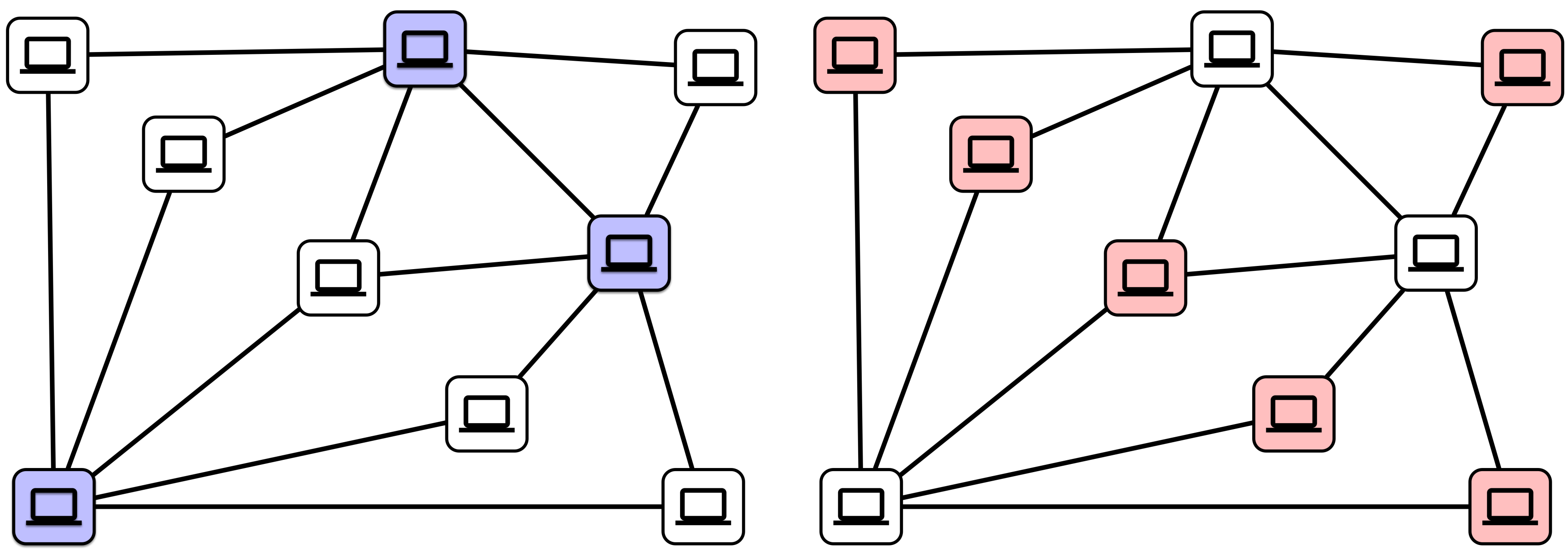
Output:

- An **independent set** $I = \{v_1, v_2, \ldots, v_k\}$
  - there is no edge between any pair of vertices in I
- I is an indpendent set of maximum size
  - there is no independent set with $\ell > k$ vertices in G

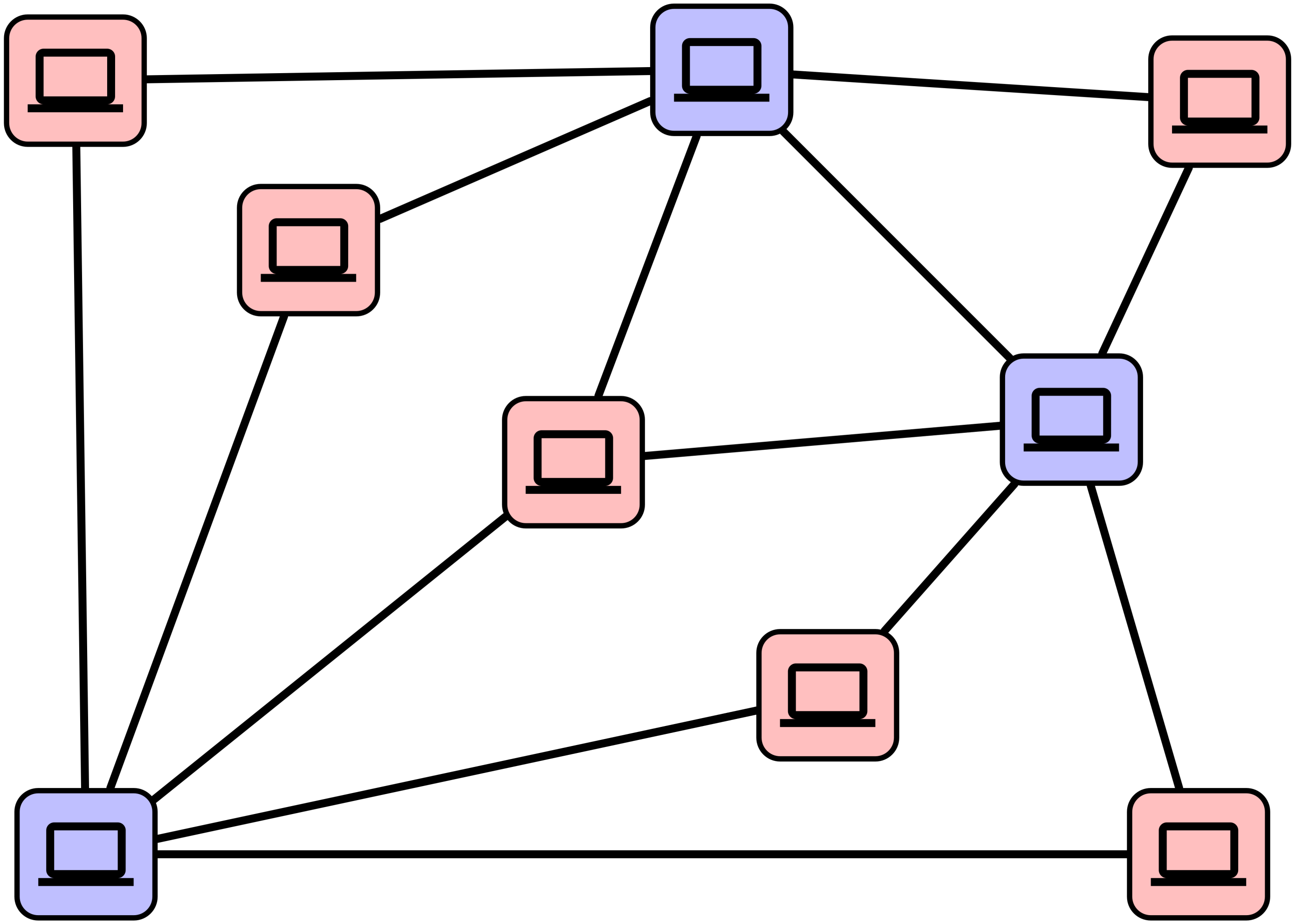
Current State

1. There is no known efficient (polynomial time) algorithm for solving MVC or MaxIS
2. There is no known proof that MVC or MaxIS cannot be solved in polynomial time

Relationship Between MVC and MaxIS?



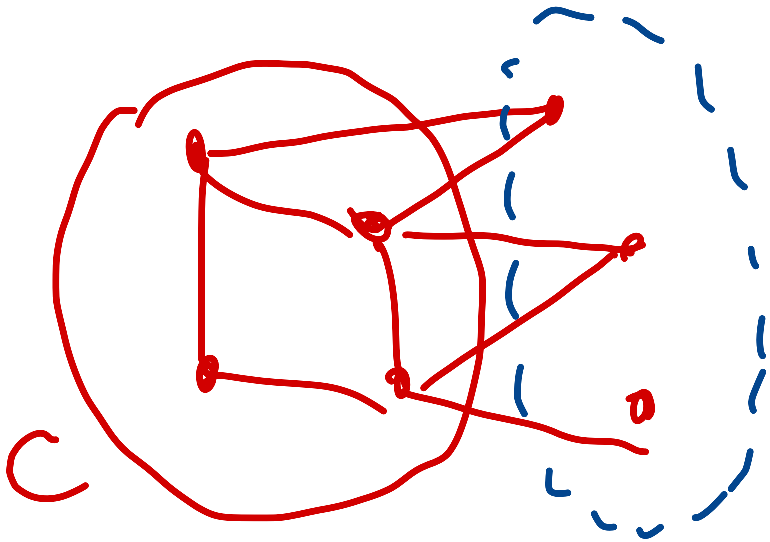
Complementary Sets



$G = (V, E)$, # vertices n

Claim 1

Suppose G has a vertex cover C of size k. Then G has an indpendent set of size $n - k$, namely $\underbrace{V - C}_{\text{elts in } V \text{ not in } C}$.



By def. of V.C. all edges have ≥ 1 endpt in C

$\Rightarrow$ no edge has both endpts in $V - C$

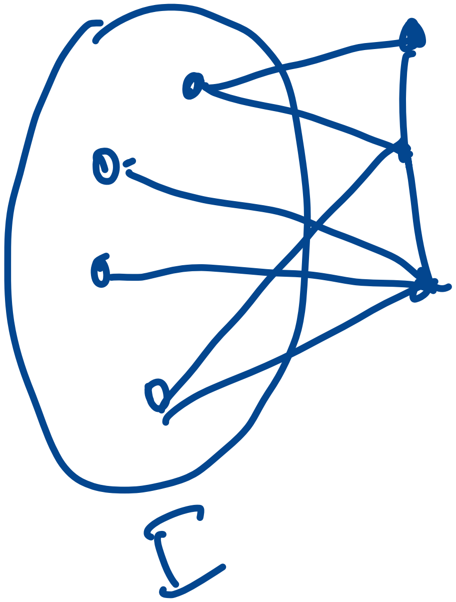
$\Rightarrow$ $V - C$ is indep. set

$k = 4$, $n = 7$

$|V - C| = n - k$

Claim 2

Suppose G has an independent set I of size k. Then G has a vertex cover of size $n - k$, namely $V - I$



By def of IS,
no edges w/ both endpts in I
$\Rightarrow$ all edges have ≥ 1 endpt in $V - I$
$\Rightarrow$ $V - I$ is V.C.

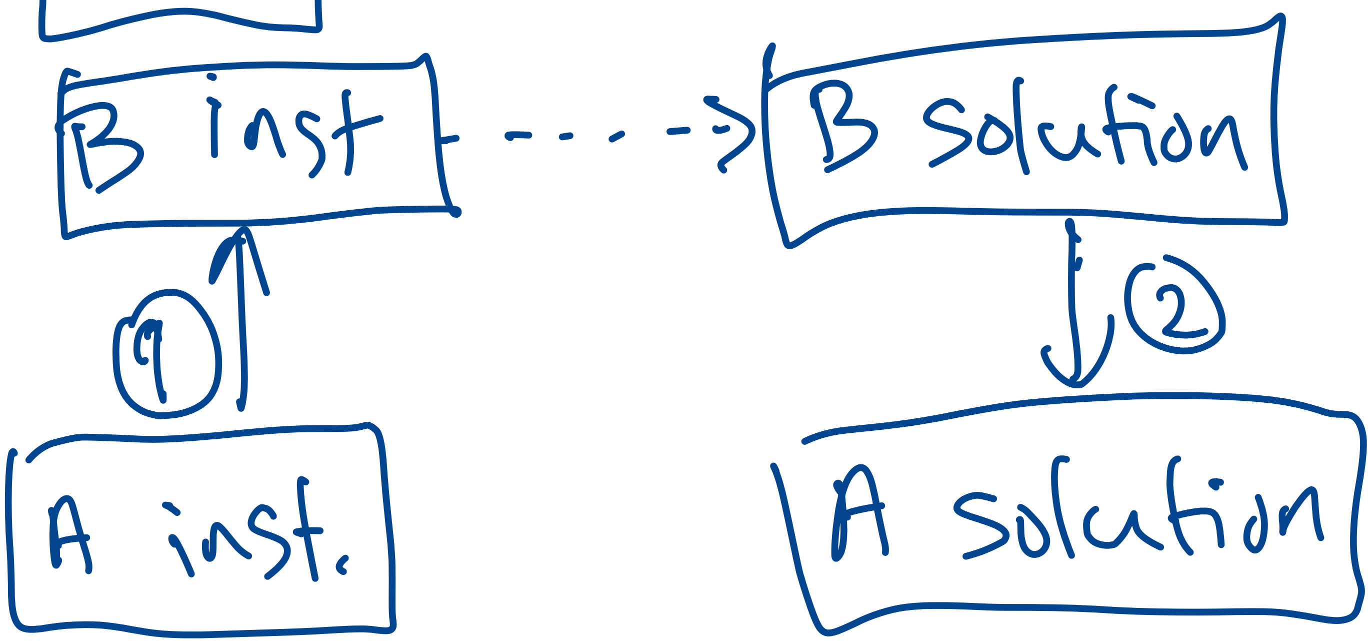
$k = 4$
$n = 7$
$|V - I| = n - k.$

Reducibility Reminder

A **polynomial time reduction** from problem A to problem B consists of

1. a polynomial time procedure to transform instances of A to instances of B
2. a polynomial time procedure to transform corresponding solutions to B to solutions to A

If there is a polynomial time reduction from A to B, then we write $\boxed{A \leq_P B.}$



Reducing MVC to MaxIS

Claim. MVC $\leq_P$ MaxIS.

Input to MVC = G (graph)

↪ interprate as input for MaxIS

Give sol'n to MaxIS, I

↪ V−I is MVC.

By Claim 1 + 2.

Reducing MaxIS to MVC

Claim. MaxIS $\leq_P$ MVC.

Same procedure

- use same input
- get MVC $= C$
- return $I = V - C$

get Max IS!

Consequences

1. If we find an efficient algorithm for MVC, then we automatically get an efficient algorithm for MaxIS
2. If we find an efficient algorithm for MaxIS, then we automatically get an efficient algorithm for MVC
3. If we prove there is no efficient algorithm for MVC, then there is no efficient algorithm for MaxIS
4. If we prove there is no efficient algorithm for MaxIS, then there is no efficient algorithm for MVC

One More Technicality

Objective. Understand *relationships* between computational problems.

Technical issue. Desired outputs for different problems can be vastly different:

- matching
- independent set
- spanning tree
- ...

Convenience. Focus on **decision problems:**

- output is “yes”/”no”

MVC vs VC

Minimum Vertex Cover (MVC)

- *Input:* Graph G
- *Output:* A *vertex cover* C of smallest possible size

Vertex Cover (VC)

Input: Graph G, number k

Output:

- "yes" if G has a vertex cover of size k
- "no" otherwise

MaxIS vs IS

Maximum Independent Set (MaxIS)

Input: Graph G

Output: an indpendent set of the largest possible size

Independent Set (IS)

Input: Graph G, number k

Output:

- "yes" if G has an indpendent set of size k
- "no" otherwise

Exercise (last HW assignment)

Given an algorithm for the decision problem, devise an algorithm for the original problem.

Reductions between VC and IS

Complexity of Decision Problems

Goal. Classify (decision) problems according to their relative *complexities:*

- which problems can be solved efficiently?
- which problems cannot be solved efficiently?
- which problems can be reduced to other problems?

Complexity Landscape

- P = decision problems that can be solved in polynomial time $O(N^c)$ (some constant c)
- EXP = decision problems can be solved in time $O(2^{N^c})$ (some constant c)

Surprising Answer

IS and VC belong to a *huge* class of natural/practical problems such that

1. none is known to admit a polynomial time algorithm
2. a polynomial time algorithm for one would imply a polynomial time algorithm for all others
3. a proof that any one cannot be solved in polynomial time would imply that none can be solved in polynomial time

The class of problems is called **NP Complete**

Next Time

- Boolean satisfiability (lecture ticket)
- Definition of NP
- NP completeness