Analysis of Algorithms

V. Adamchik

CSCI 570

Lecture 12

University of Southern California

Spring 2023

NP-Completeness - II

Reading: chapter 9

In 1936 Alan Turing described:

- A simple formal model of computation now known as Turing machines.
- A proof that TM can NOT solve the halting problem.
- A proof that NO Turing machine can determine whether a *given* proposition is provable from the axioms of first-order logic.
- Compelling arguments that a problem not computable by a Turing machine is not "computable" in the absolute (human) sense.
- A non-deterministic Turing machine: for each state it makes an arbitrary choice between a finite of possible transitions.

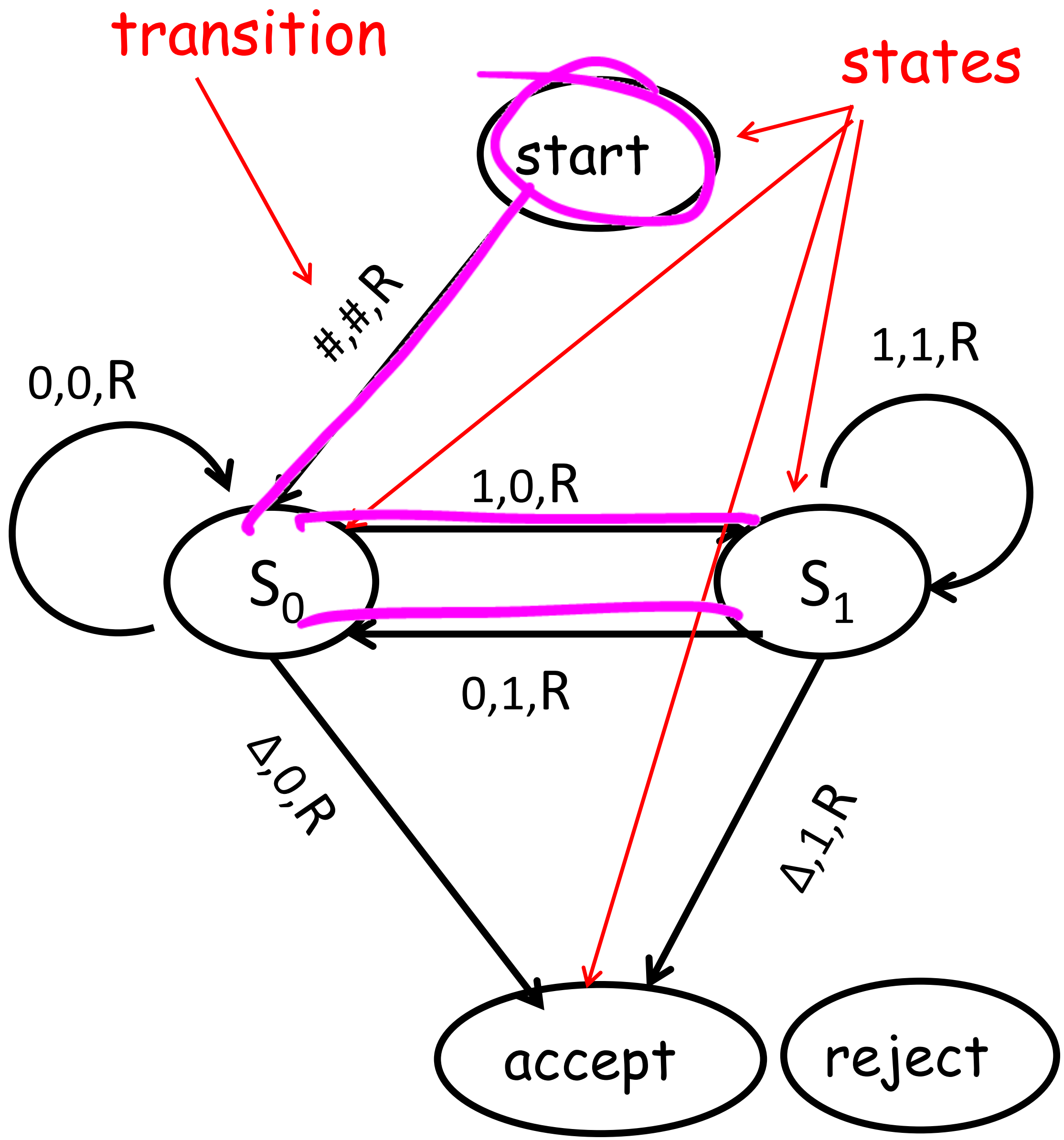
Deterministic Turing Machine

The machine that takes a binary string and appends 0 to the left side of the string.

Input: #10010Δ
Output: #010010Δ

- leftmost char
Δ – rightmost char

Transition on each edge
read,write,move (L or R)



Non-Deterministic Turing Machine

- NDTM is a choice machine: for each state it makes an arbitrary choice between a finite (possibly zero) number of states.
- The computation of a NDTM is a tree of possible configuration paths.
- One way to visualize NDTM is that it makes an exact copy of itself for each available transition, and each machine continues the computation.
- Rabin & Scott in 1959 shown that adding non-determinism does not result in more powerful machine.
- For any NDTM, there is a DTM that accepts and rejects exactly the same strings as NDTM.
- P vs. NP is about whether we can simulate NDTM in polynomial time.

Complexity Classes

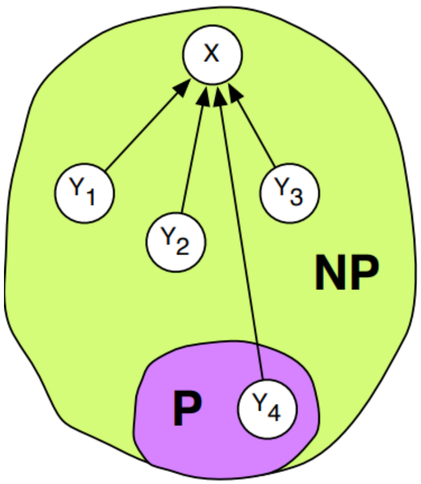
P = set of problems that can be solved in polynomial time by a DTM.

NP = set of problems that can be solved in polynomial time by a NDTM.

NP = set of problems for which solution can be verified in polynomial time by a deterministic TM.

X is *NP-Hard*, if $\forall Y \in$ NP and $Y \leq_p X$.

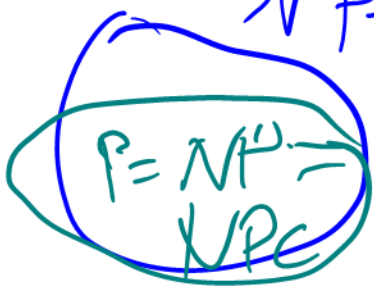
X is *NP-Complete*,
if X is NP-Hard and $X \in$ NP.

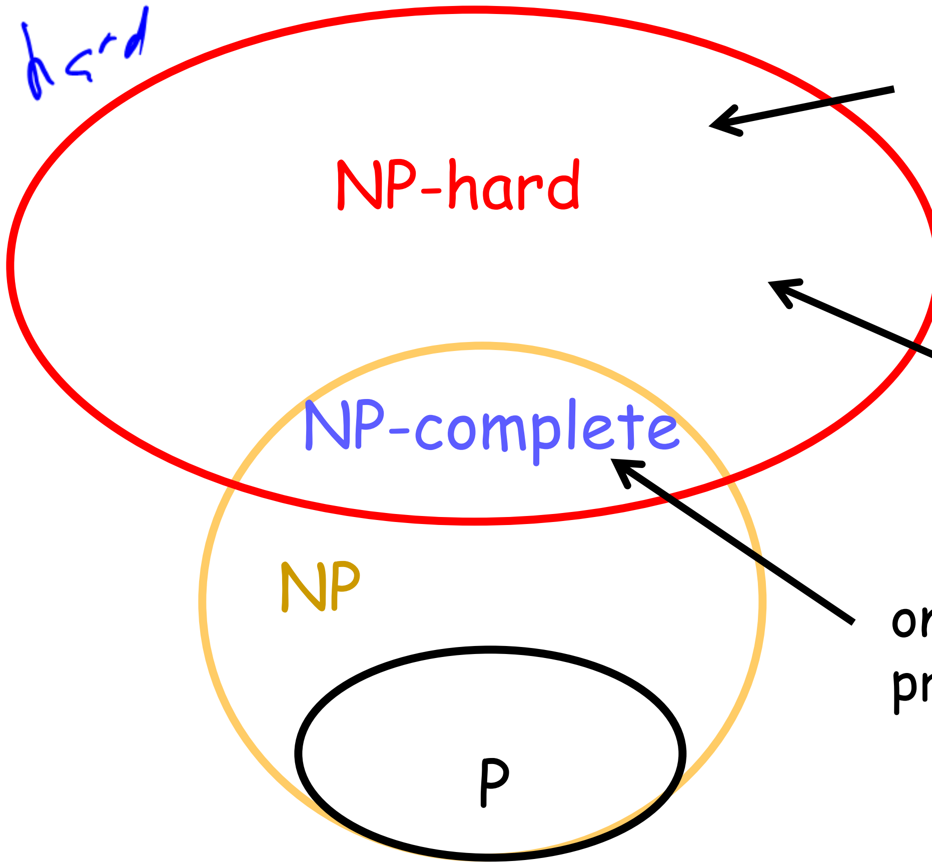


Venn Diagram ($P \neq NP$)

P = NP

NP-hard





Undecidable:
Halting Problem

optimization &
decision
problems

only decision
problems

NPC problems are the most difficult NP problems.

It's not known if NPC problems can be solved by a *deterministic* TM in polynomial time.

NPC problems can be solved by a *non-deterministic* TM in polynomial time.

NP-Complete Problems

Cook-Levin Theorem: CNF SAT is NP-complete.

Independent Set:
Given graph G and a number k, does G contain a set of at least k independent vertices?

Vertex Cover:
Given a graph G and a number k, does G contain a vertex cover of size at most k.

A Hamiltonian cycle:
Given a graph G, does G contain a cycle that visits each vertex exactly once.

NP-Completeness Proof Method

To show that X is NP-Complete:

1) Show that X is in NP
2) Pick a problem Y, known to be an NP-Complete
3) Prove Y $\leq_p$ X (reduce Y to X)

In lecture 11 we have proved that Independent Set is NP-Complete by reduction from 3-SAT (3SAT $\leq_p$ IndSet)

Reduction from 3SAT to IndSet consists of three parts:

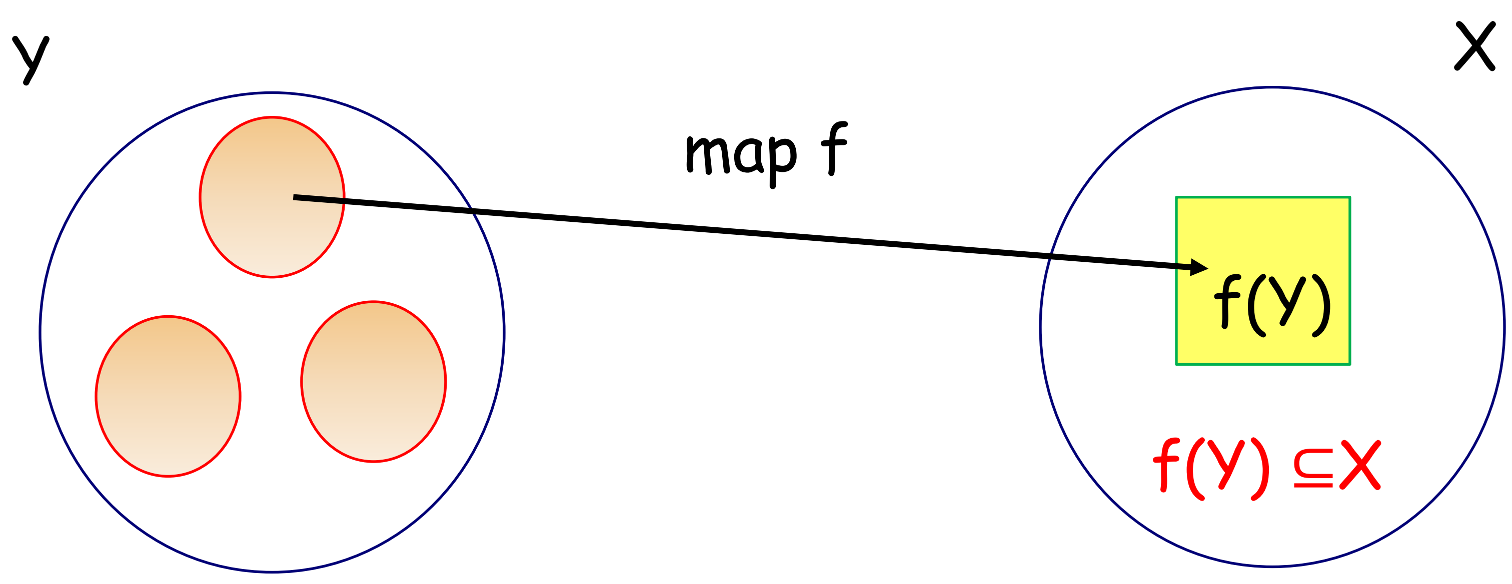
- we transform an arbitrary CNF formula into a special graph G and a specific integer k, in polynomial time.
- we transform an arbitrary satisfying assignment for 3SAT into an independent set in G of size k.
- we transform an arbitrary independent set (in G) of size k into a satisfying assignment for 3SAT.

The confusing point is that the reduction $Y \leq_p X$ only "works one way", but the correctness proof needs to "work both ways".

The correctness proofs are not actually symmetric.

The proof needs to handle arbitrary instances of Y, but only needs to handle the special instances of X produced by the reduction.

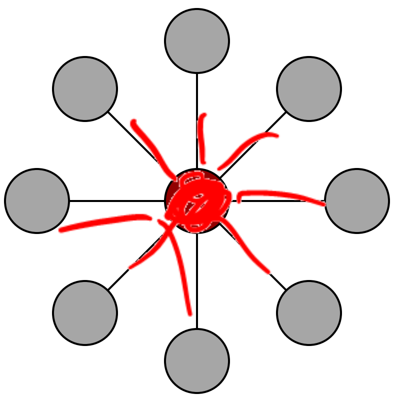
This asymmetry is the key to understanding reductions.



Vertex Cover

Given G=(V,E), find the smallest S⊂V s.t. every edge is incident to vertices in S.

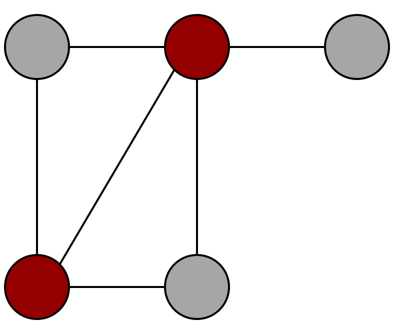
OPTIMIZATION



The minimum vertex cover problem, MinVC, asks for the size of the smallest vertex cover in a given graph.

Decision Problem:

Given G and K, does G contain a VC of size |VC| = K?



Vertex Cover

<u>Theorem</u>: *for a graph $G=(V,E)$, S is an independent set if and only if $V-S$ is a vertex cover*

Proof. $\Rightarrow$)

Vertex Cover

<u>Theorem</u>: *for a graph $G=(V,E)$, S is a independent set if and only if $V-S$ is a vertex cover*

Proof. $\Leftarrow$)

Min Vertex Cover in NP-Hard

MaxIndSet $\leq_p$ MinVC

By the previous theorem.

Vertex Cover in NP-Complete

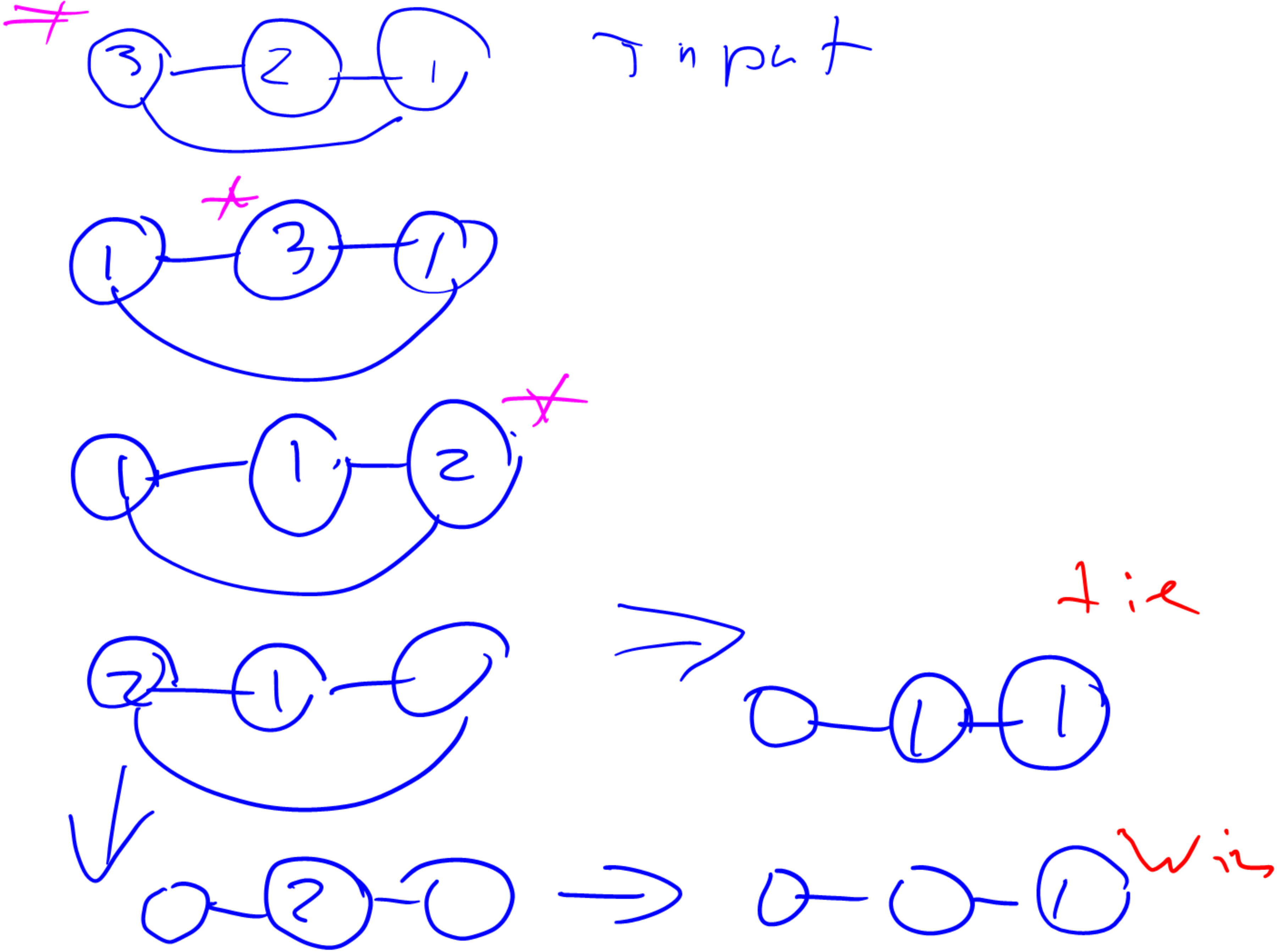
Claim: *a graph G=(V,E) has an independent set of size at least k if and only if G has a vertex cover of size at most V-k.*

Ind. Set $\leq_p$ Vertex Cover

By the previous theorem.

Discussion Problem 1

You are given an undirected graph G = (V, E) and for each vertex v, you are given a number p(v) that denotes the number of pebbles placed on v. We will now play a game where the following move is the only move allowed. You can pick a vertex u that contains at least two pebbles, and remove two pebbles from u and add one pebble to an adjacent vertex. The objective of the game is to perform a sequence of moves such that we are left with exactly one pebble in the whole graph. Show that the problem of deciding if we can reach the objective is NP-complete. Reduce from the Hamiltonian Path problem.


Input
tie
Win

① Pebbles $\in NP$

② $HP \leq_p$ Pebbles

Construction:

HP(G)

Claim: G has a HP $\Leftrightarrow$ G' has a wining sequence.

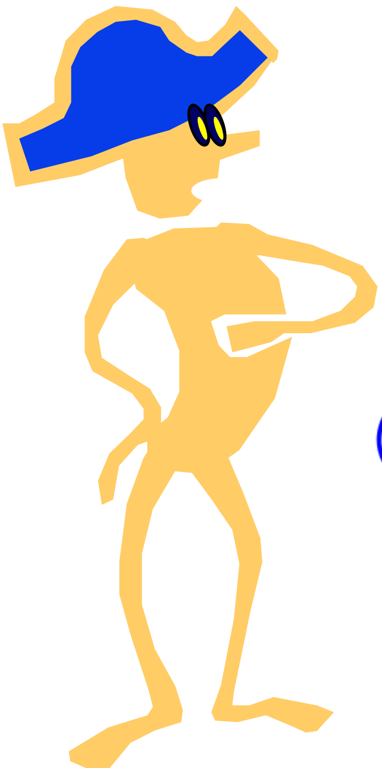
$\Rightarrow$ G has a HP

Goal! to find the winning path

Decision: follow the HP path

$\Leftarrow$ G' has a winning path

Yes, we won't get a HP



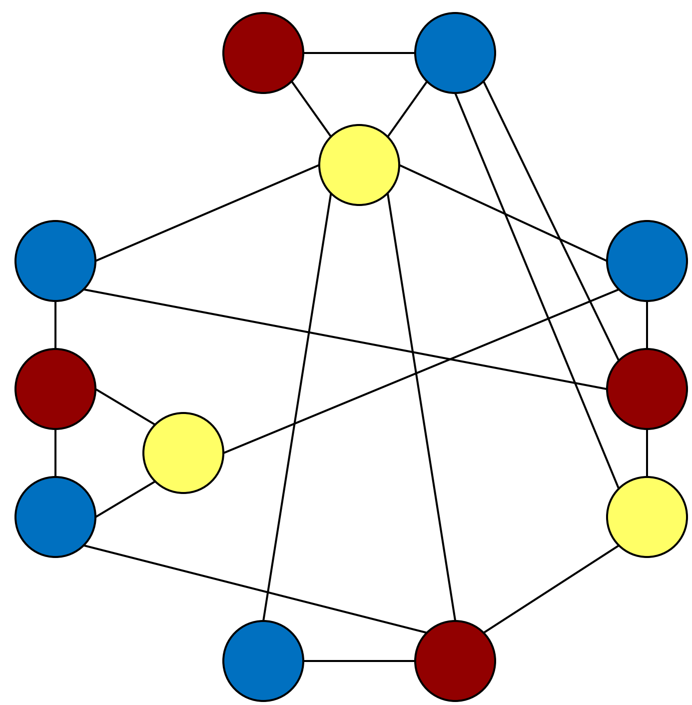
Graph Coloring

Given a graph, can you color the nodes with ≤ k colors such that the endpoints of every edge are colored differently?

$k \le 4$

① Planar Graphs $\in P$

Theorem. (k>2)
k-Coloring is NP-complete.



Graph Coloring: k = 2

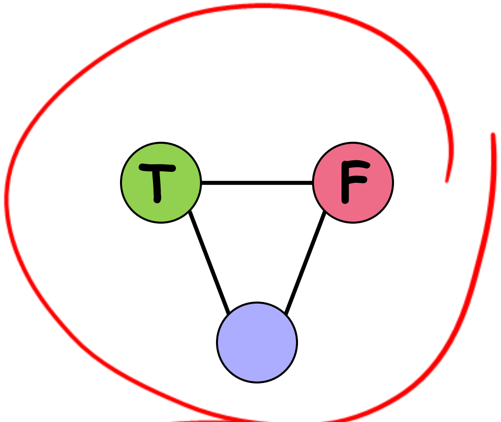
How can we test if a graph has a 2-coloring?

3-SAT $\leq_p$ 3-colorable

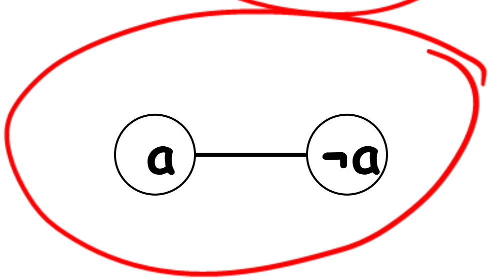
We construct a graph G that will be 3-colorable iff the 3-SAT instance is satisfiable.

Graph G consists of the following gadgets.

A truth gadget:



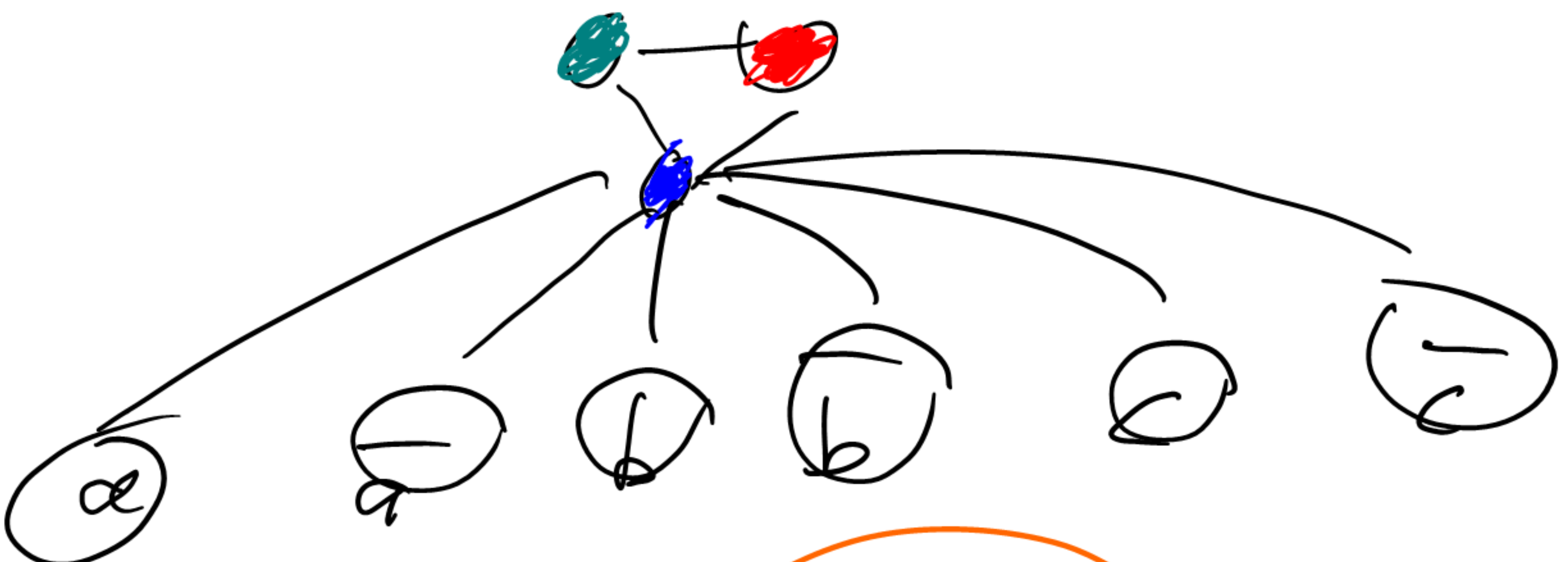
A gadget for each variable:



3-SAT $\leq_p$ 3-colorable

Combining those gadgets together (for three literals)

a, b, c



clause gadget ?

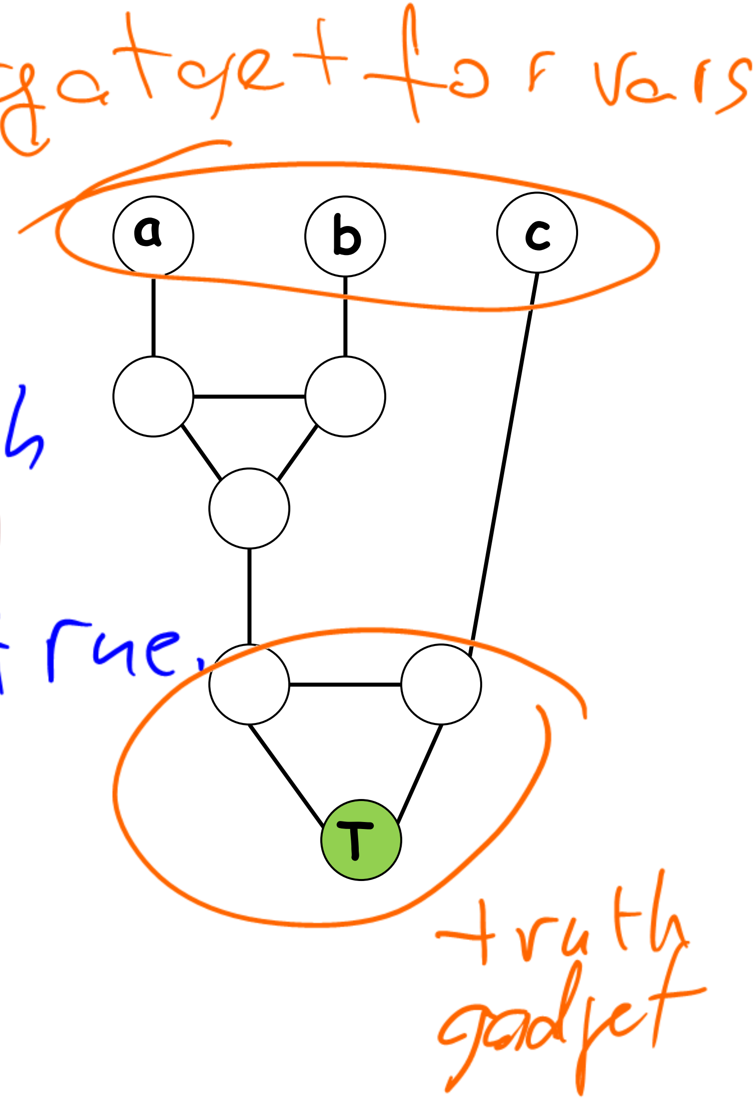
3-SAT $\leq_p$ 3-colorable

A special gadget for each clause

This gadget connects a truth gadget with variable gadgets.

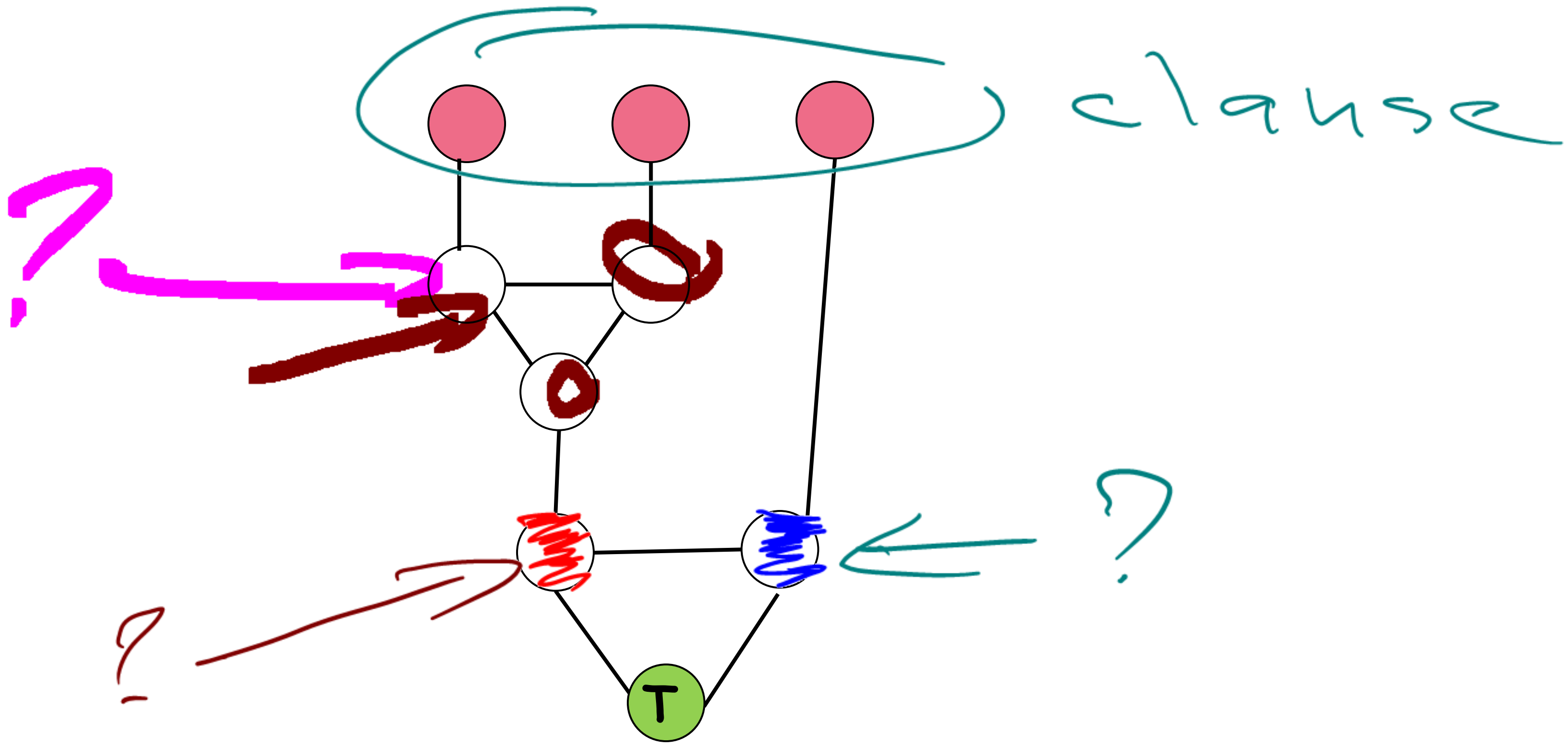
we can color this graph with 3 colors iff one of the colors is true.

Assume
$a = b = c = F$



3-SAT $\leq_p$ 3-colorable

Suppose all a, b and c are all False (red).

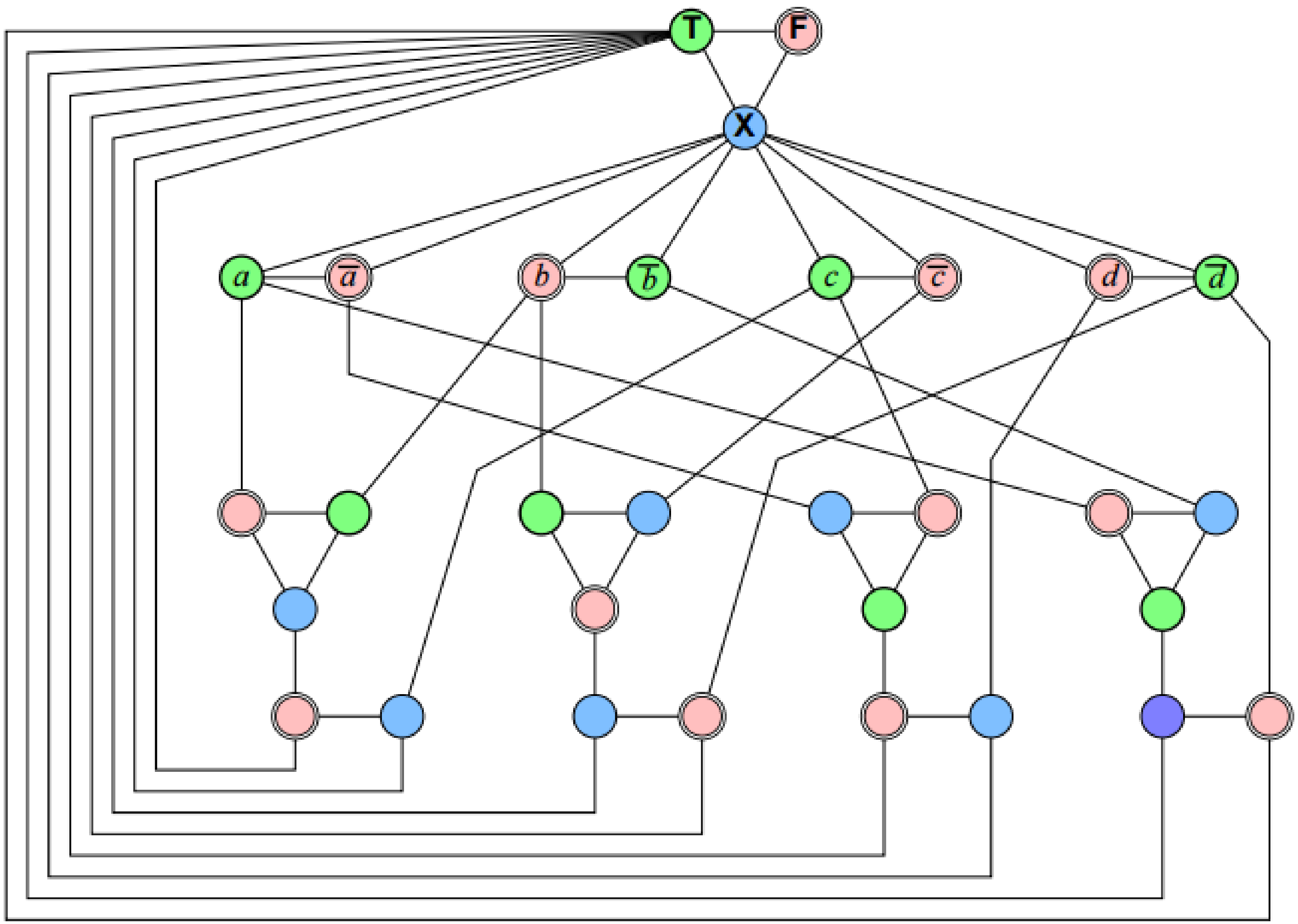


3-SAT $\leq_p$ 3-colorable

We have showed that if all the variables in a clause are false, the gadget cannot be 3-colored.

Example: $a \vee \neg b \vee c$

Example with four clauses



a=c=T

b=d=F

A 3-colorable graph derived from a satisfiable 3CNF formula.

$$(a \vee b \vee c) \wedge (b \vee \bar{c} \vee \bar{d}) \wedge (\bar{a} \vee c \vee d) \wedge (a \vee \bar{b} \vee \bar{d})$$

3-SAT $\leq_p$ 3-colorable

Claim: *3-SAT instance is satisfiable if and only if G is 3-colorable.*

Given: satisfiable 3SAT

Proof: ⇒)

Goal: truth assignment

construction

truth assignment — color

variable assignment — color

clause gadget — coloring is forced

3-SAT $\leq_p$ 3-colorable

Claim: *3-SAT instance is satisfiable if and only if G is 3-colorable.*

Proof: $\Leftarrow$) Given: a special graph, which is 3-colorable

Goal: find a truth assignment!

Start with variable gadget...

Sudoku: $n^2 \times n^2$

$n \to \infty$

2			3		8		5	
		3		4	5	9	8	
		8			9	7	3	4
6		7		9				
9	8						1	7
				5		6		9
3	1	9	7			2		
	4	6	5	2		8		
	2		9		3			1

NP-?

NP-hard?

9-colors

Sudoku graph:

vertex: 81

edges:

810

810

Sudoku Graph

2			3		8		5	
		3		4	5	9	8	
		8			9	7	3	4
6		7		9				
9	8						1	7
				5		6		9
3	1	9	7			2		
	4	6	5	2		8		
	2		9		3			1

Sudoku

Constructing a Sudoku graph, we have proved:

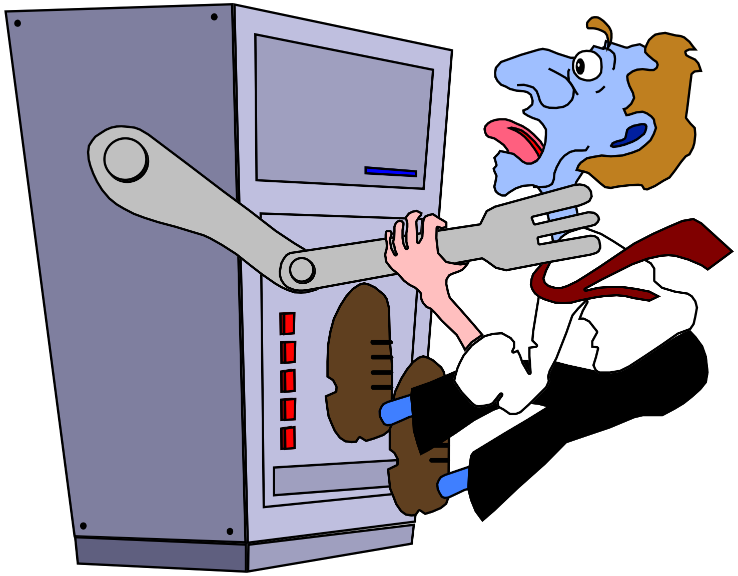
① Sudoku $\leq_p$ 9-colors $\in$ NPC

this does not prove that Sudoku is NPC

② how can we use this reduction?

Sudoku $\leq_p$ 9-colors

Don't be afraid of NP-hard problems.



Many reasonable instances (of practical interest) of problems in class NP can be solved!

The largest solved TSP an 85,900-vertex route calculated in 2006. The graph corresponds to the design of a customized computer chip created at Bell Laboratories, and the solution exhibits the shortest path for a laser to follow as it sculpts the chip.

570

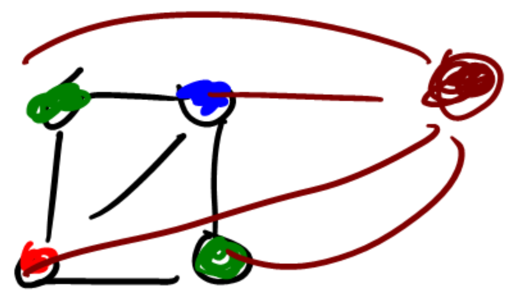
Discussion Problem 2

Prove that 4-COLOR is NP-complete.

3-color $\leq_p$ 4-color

G

f'



claim: k-colors $\leq_p$ (k+1)-colors