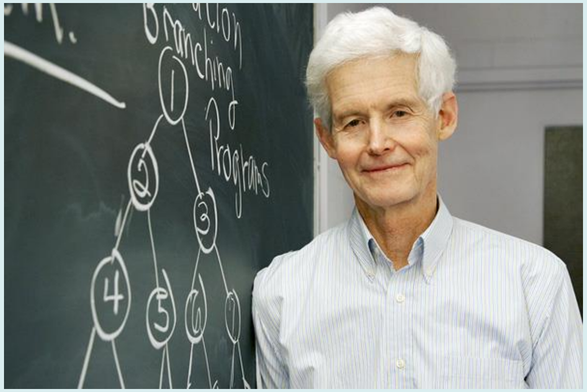
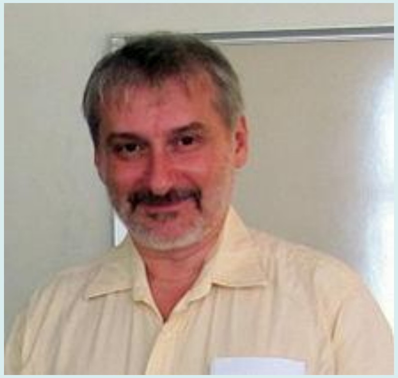






# CSE 417
# Algorithms and Complexity

Autumn 2023

Lecture 28

NP-Completeness

# Announcements

- Homework 9
- Exam practice problems on course homepage
- Final Exam: Monday, December 11, 8:30 AM
  – One Hour Fifty Minutes

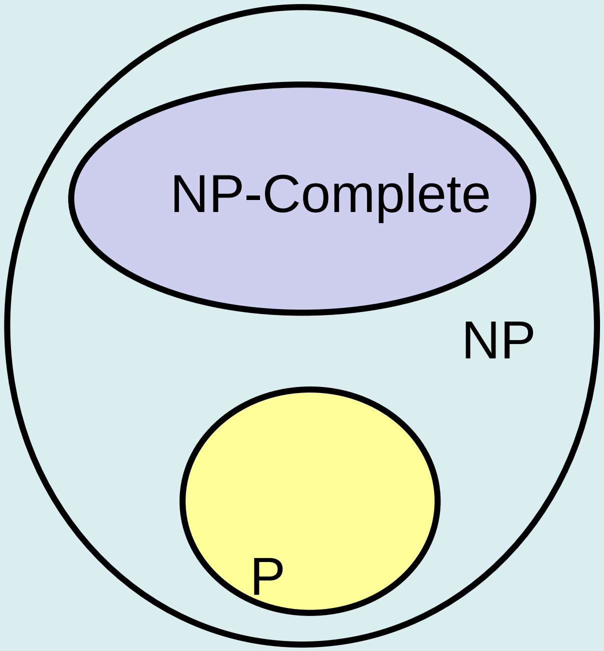
| | |
|---|---|
| ~~Fri, Dec 1~~ | ~~Net Flow Applications~~ |
| ~~Mon, Dec 4~~ | ~~Net Flow Applications + NP-Completeness~~ |
| Wed, Dec 6 | NP-Completeness |
| Fri, Dec 8 | NP-Completeness |
| Mon, Dec 11 | Final Exam |

# Key Idea: Problem Reduction

- Use an algorithm for problem Y to solve problem X.
  - This means that problem Y is more difficult that problem X
- Terminology: X is reducible to Y
  - Notation: $X <_P Y$

# The Universe

- P:  Polynomial Time
- NP: Nondeterministic Polynomial Time
  – Problems where a “yes” answer can be verified in polynomial time
- NP-Complete
  – The hardest problems in NP



# Polynomial time reductions

- X is Polynomial Time Reducible to Y
  - Solve problem X with a polynomial number of computation steps and a polynomial number of calls to a black box that solves Y
  - Notations: $X <_P Y$

- Usually, this is converting an input of X to an input for Y, solving Y, and then converting the answer back

# Composability Lemma

- If $X <_P Y$ and $Y <_P Z$ then $X <_P Z$

# Lemmas

- Suppose $X <_P Y$. If Y can be solved in polynomial time, then X can be solved in polynomial time.
- Suppose $X <_P Y$. If X cannot be solved in polynomial time, then Y cannot be solved in polynomial time.

# NP-Completeness

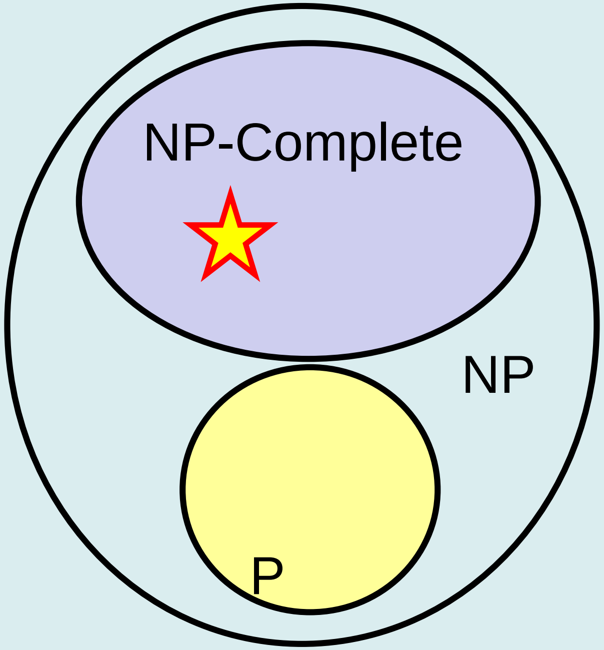
- A problem X is NP-complete if
  - X is in NP
  - For every Y in NP, $Y <_P X$

- X is a “hardest” problem in NP

- If X is NP-Complete, Z is in NP and $X <_P Z$
  - Then Z is NP-Complete

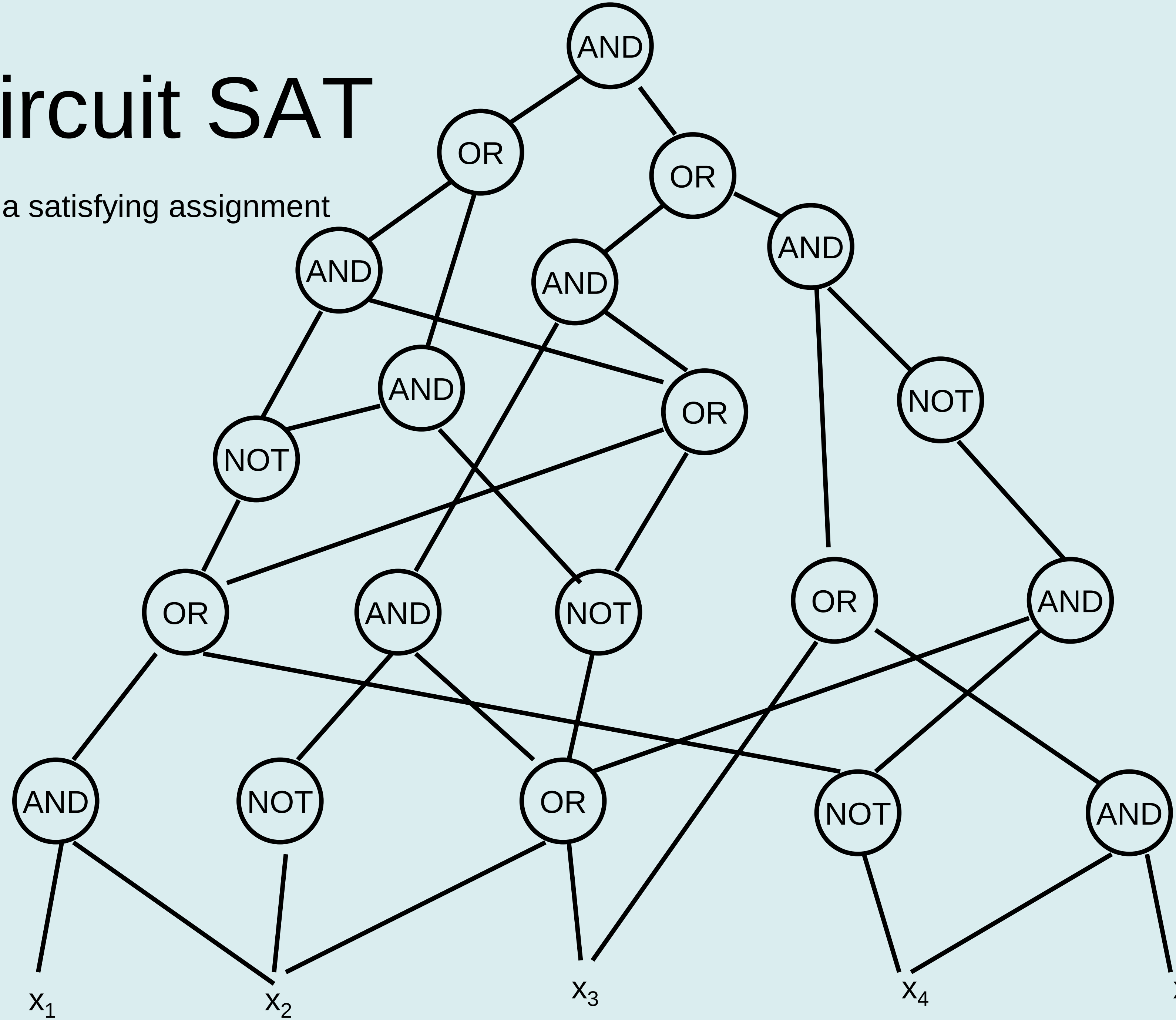
# Cook's Theorem

- There is an NP Complete problem
  - The Circuit Satisfiability Problem

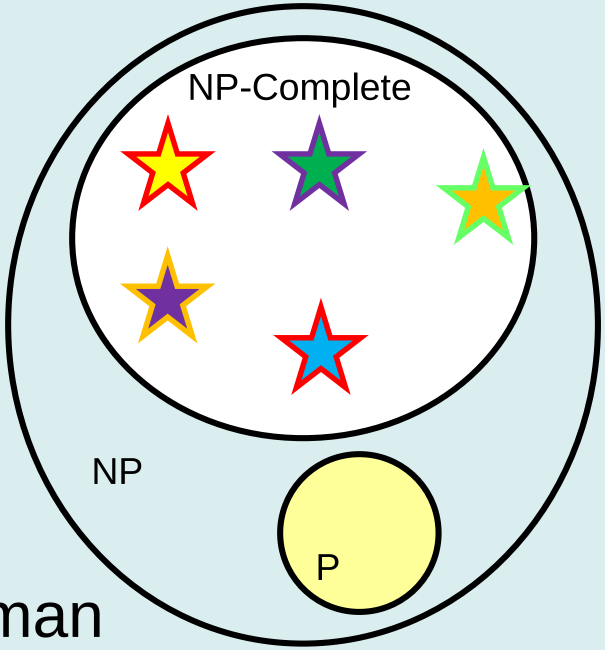


# Circuit SAT

Find a satisfying assignment



# Populating the NP-Completeness Universe



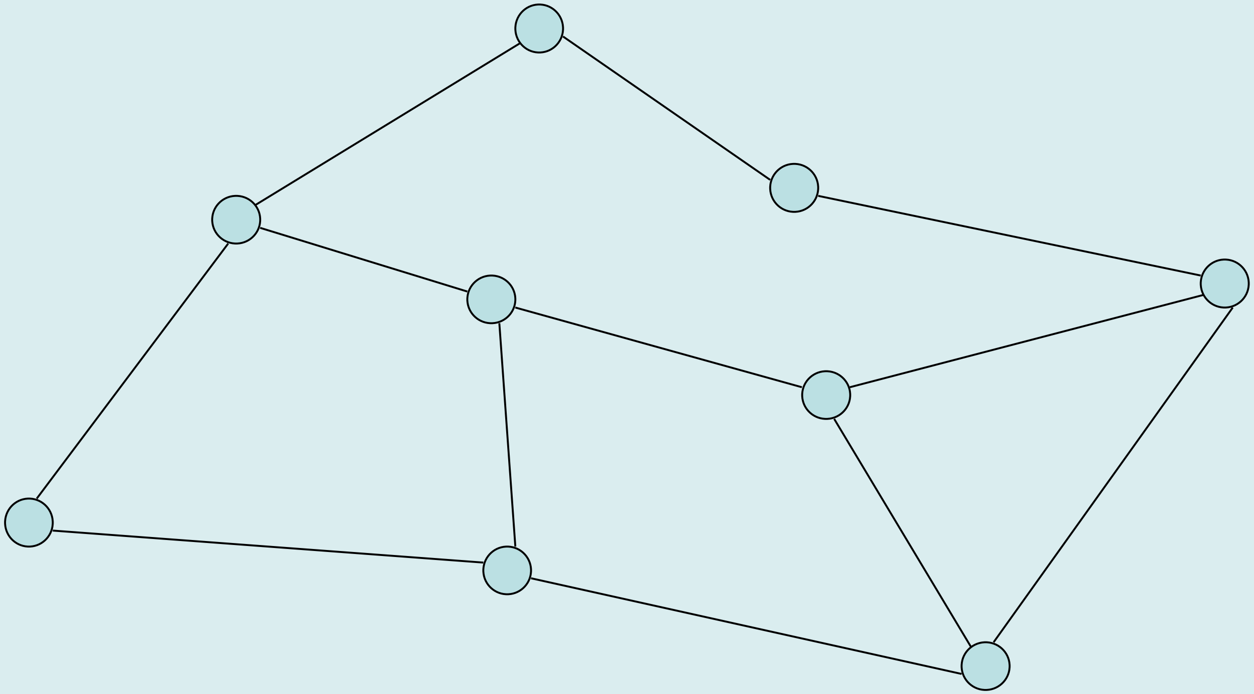

- Circuit Sat $<_P$ 3-SAT
- 3-SAT $<_P$ Independent Set
- 3-SAT $<_P$ Vertex Cover
- Independent Set $<_P$ Clique
- 3-SAT $<_P$ Hamiltonian Circuit
- Hamiltonian Circuit $<_P$ Traveling Salesman
- 3-SAT $<_P$ Integer Linear Programming
- 3-SAT $<_P$ Graph Coloring
- 3-SAT $<_P$ Subset Sum
- Subset Sum $<_P$ Scheduling with Release times and deadlines

# NP Completeness Proofs

- If X is NP-Complete, Z is in NP and X $<_P$ Z
  - Then Z is NP-Complete

# Graph Coloring

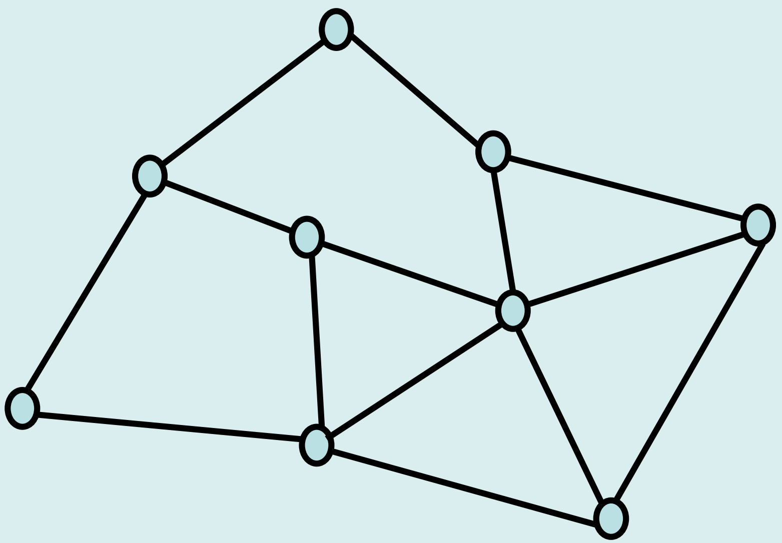
- NP-Complete
  - Graph 3-coloring
- Polynomial
  - Graph 2-Coloring

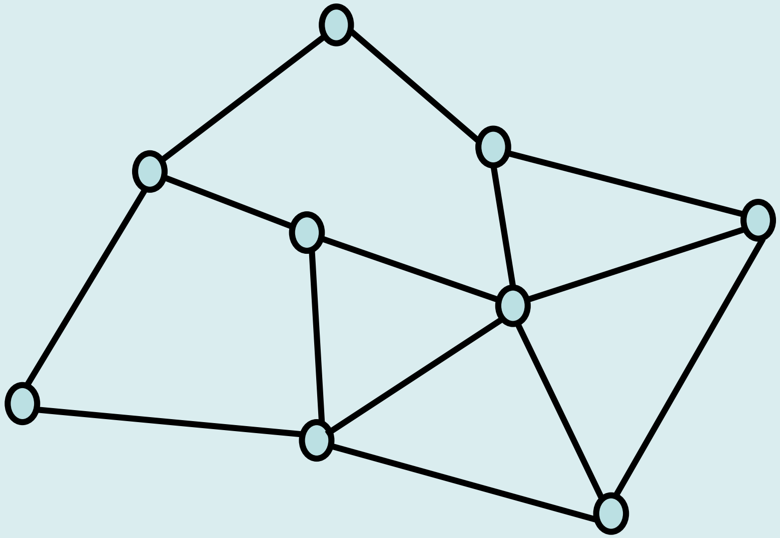


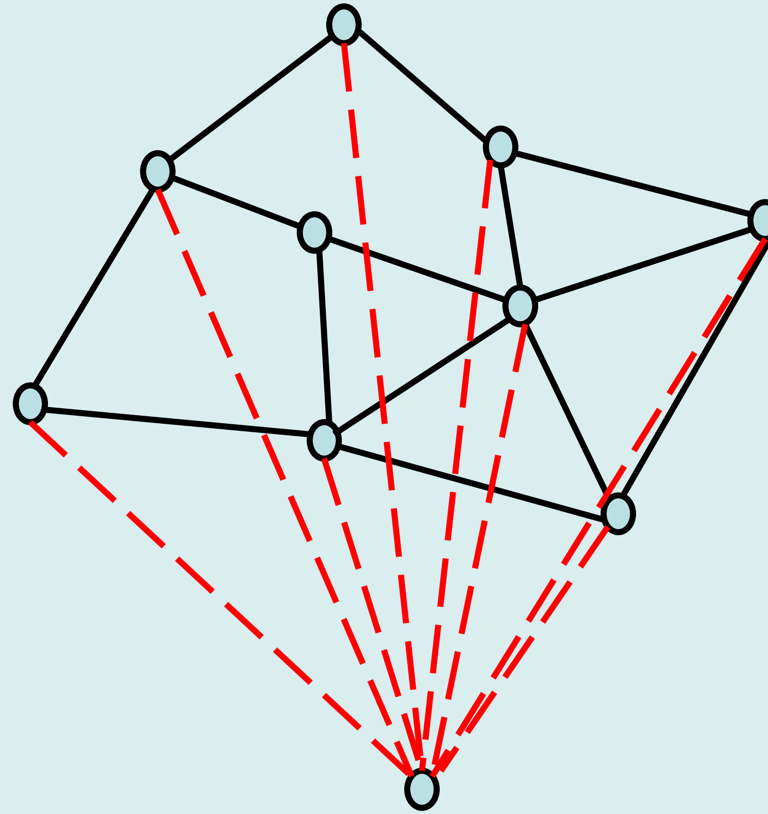
# Graph 4-Coloring

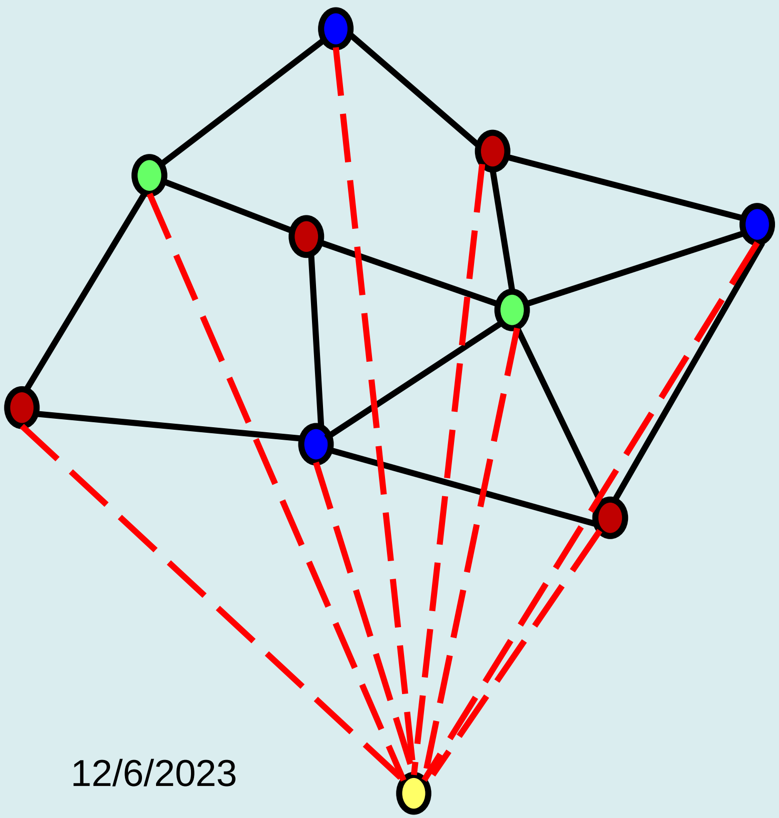
- Given a graph G,  can G be colored with 4 colors?
- Prove 4-Coloring is NP Complete
- Proof:  3-Coloring $<_P$ 4-Coloring
- Show that you can 3-Color a graph if you have an algorithm to 4-Color a graph

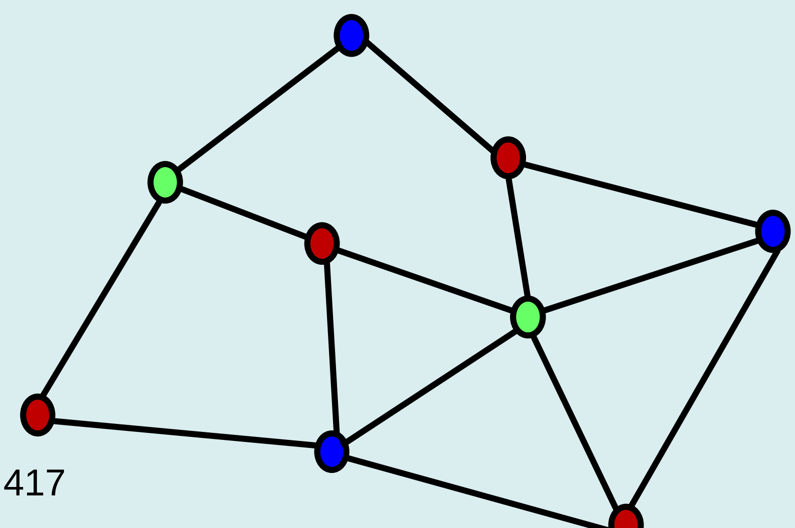
# 3-Coloring $<_P$ 4-Coloring



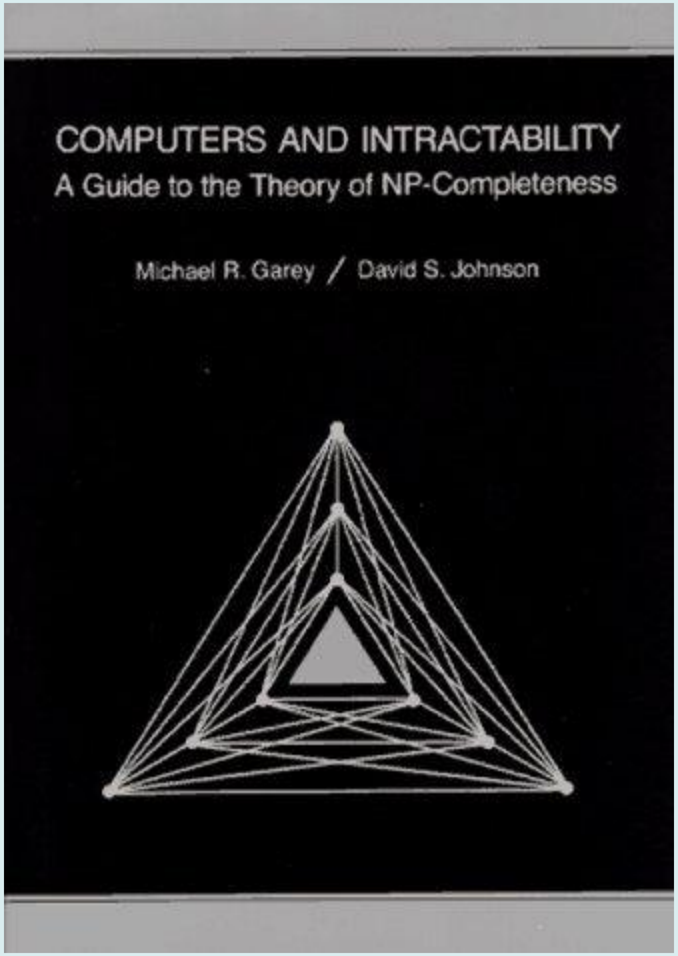




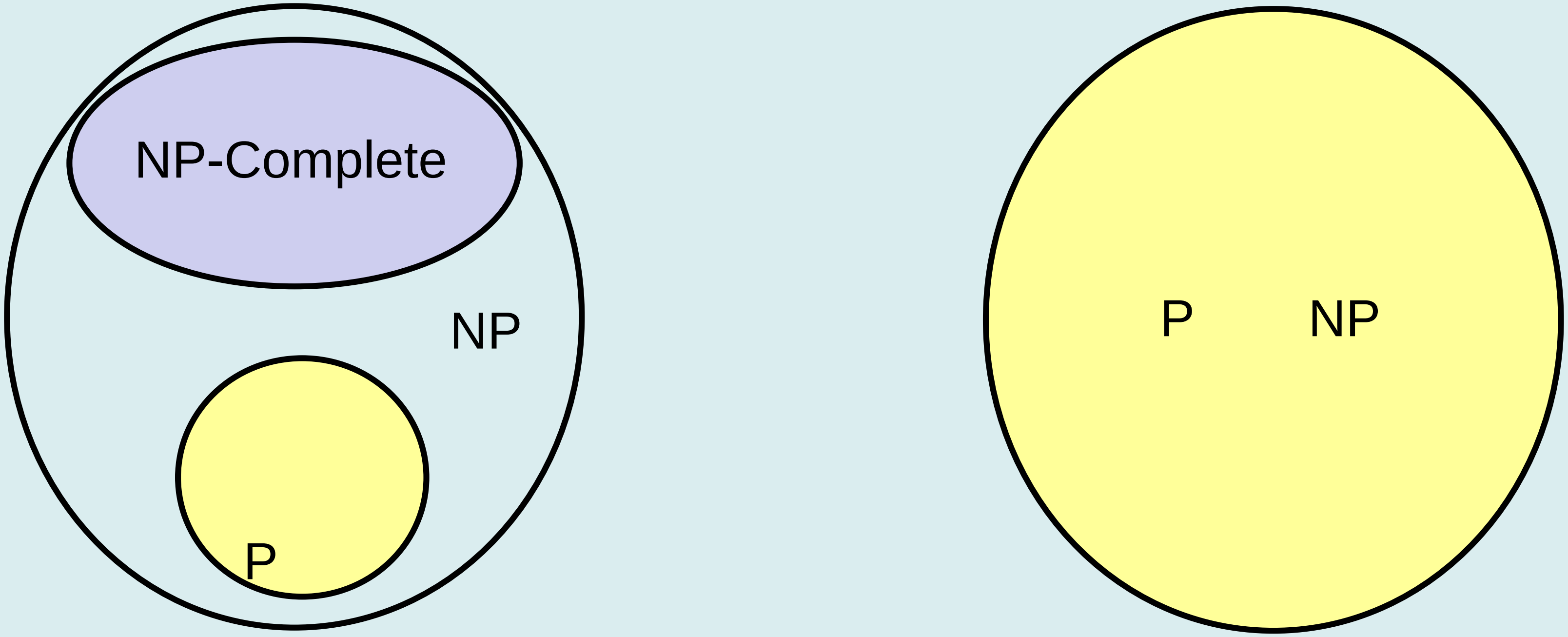




# Garey and Johnson



# P vs. NP Question



# How to prove P = NP

If X is NP-Complete and X can be solved in polynomial time, then P = NP

# Satisfiability

Literal: A Boolean variable or its negation. $x_i$ or $\overline{x_i}$

Clause: A disjunction of literals. $C_j = x_1 \vee \overline{x_2} \vee x_3$

Conjunctive normal form: A propositional formula $\Phi$ that is the conjunction of clauses. $\Phi = C_1 \wedge C_2 \wedge C_3 \wedge C_4$

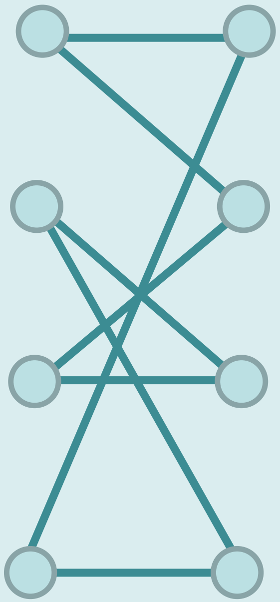
SAT: Given CNF formula $\Phi$, does it have a satisfying truth assignment?

3-SAT: SAT where each clause contains exactly 3 literals.

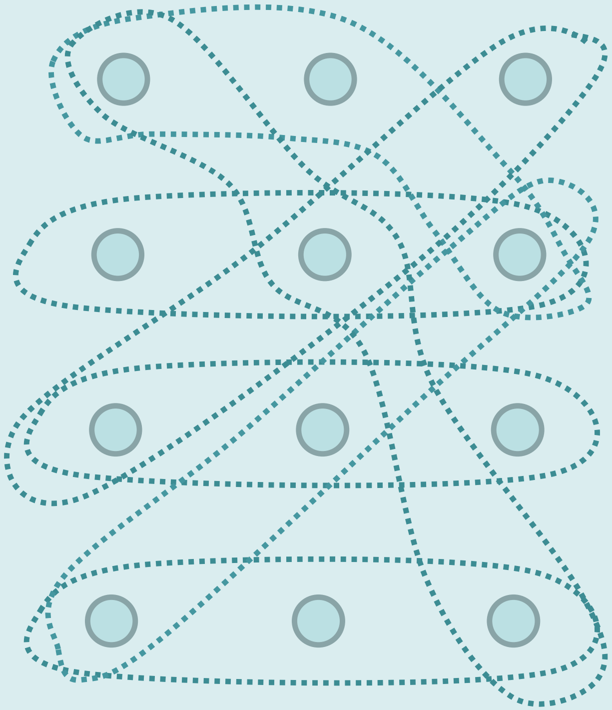
Ex: $\left(\overline{x_1} \vee x_2 \vee x_3\right) \wedge \left(x_1 \vee \overline{x_2} \vee x_3\right) \wedge \left(x_2 \vee x_3\right) \wedge \left(\overline{x_1} \vee \overline{x_2} \vee \overline{x_3}\right)$

Yes: $x_1$ = true, $x_2$ = true $x_3$ = false.

# Matching

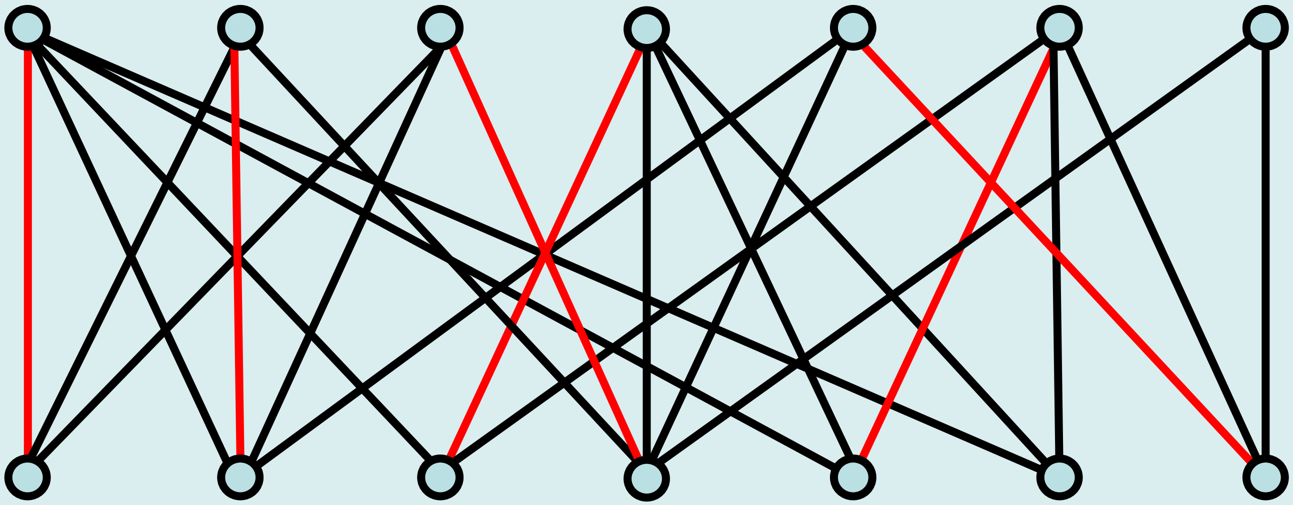


Two dimensional matching



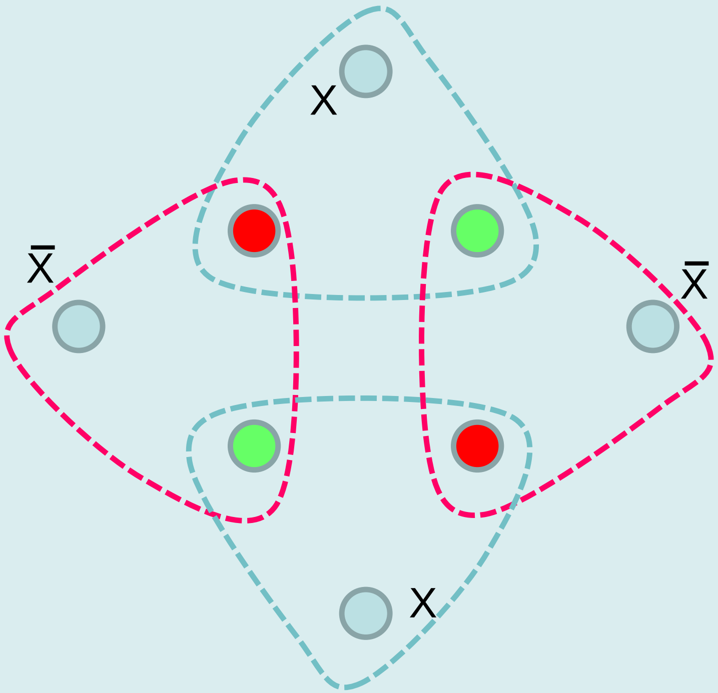
Three dimensional matching (3DM)

# Augmenting Path Algorithm for Matching

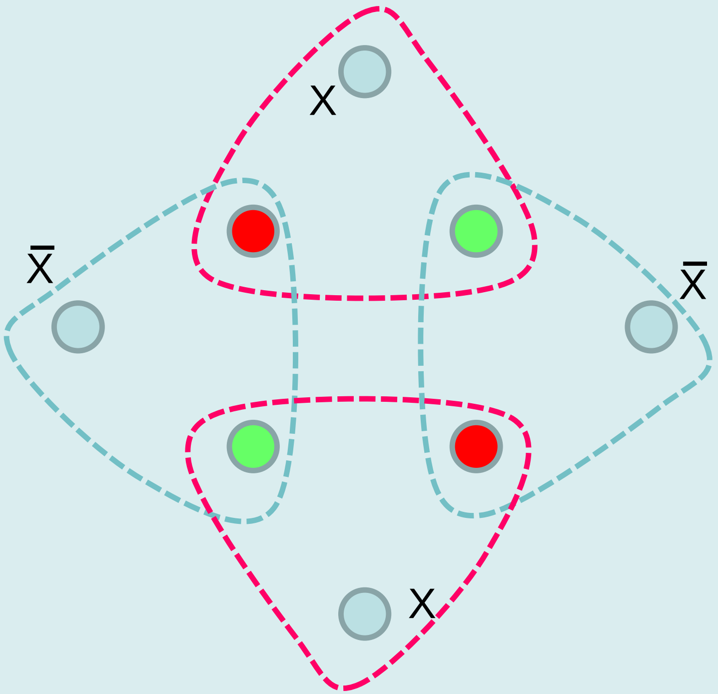


Find augmenting path in O(m) time
n phases of augmentation
O(nm) time algorithm for matching

# 3-SAT $<_P$ 3DM



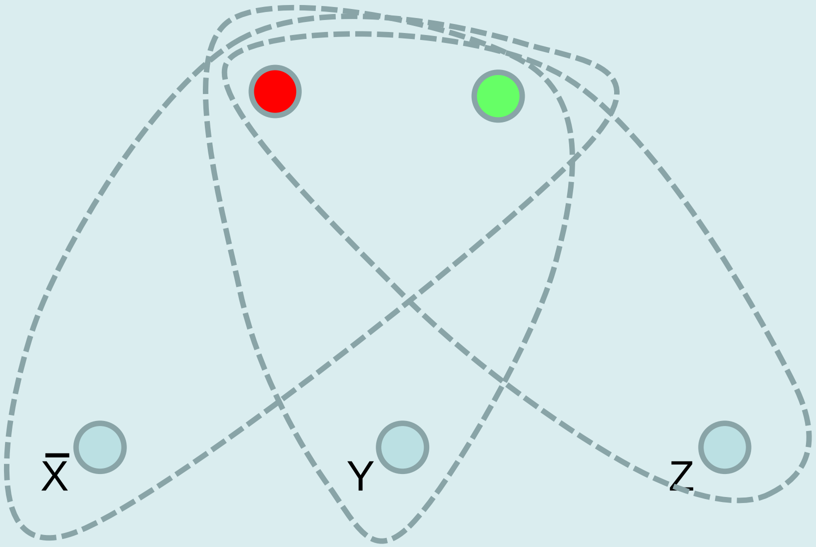

X True



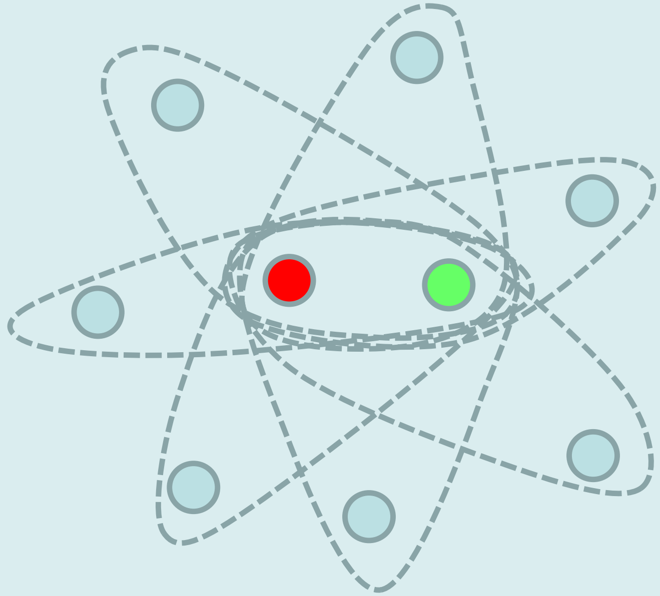

X False

## Truth Setting Gadget

# 3-SAT $<_P$ 3DM




Clause gadget for ($\bar{X}$ OR Y OR Z)



Garbage Collection Gadget
(Many copies)

# Exact Cover (sets of size 3) XC3

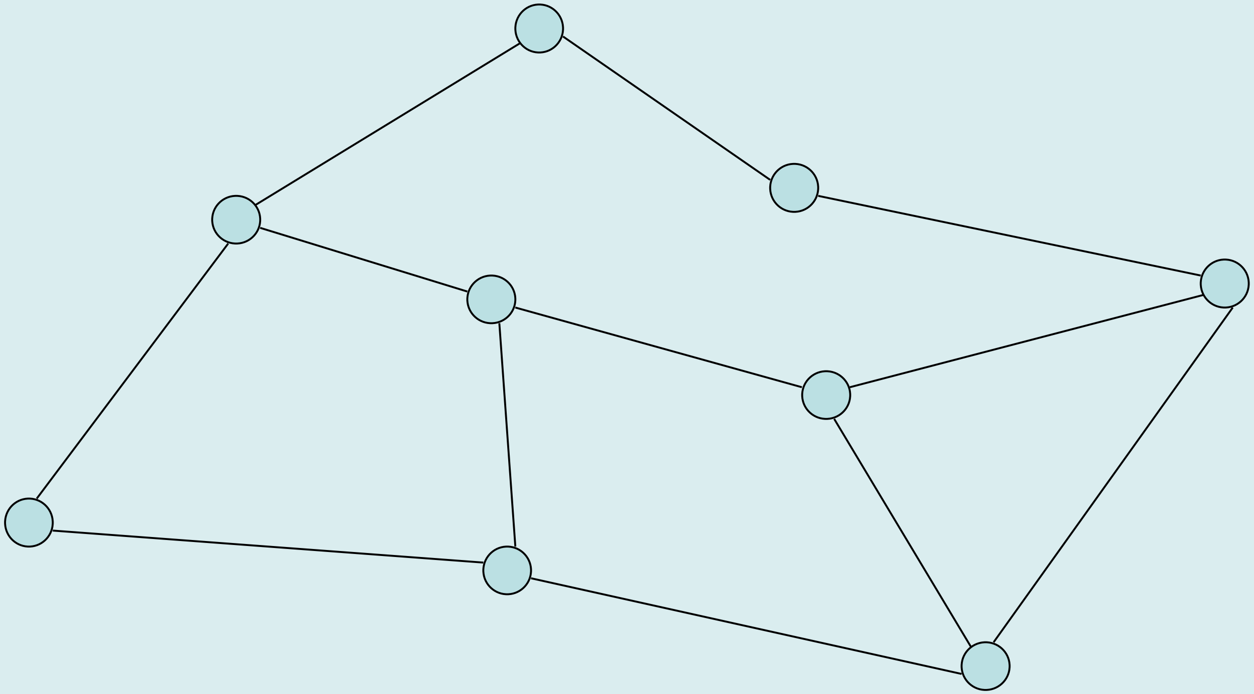
Given a collection of sets of size 3 of a domain of size 3N, is there a sub-collection of N sets that cover the sets

(A, B, C), (D, E, F), (A, B, G),
(A, C, I), (B, E, G), (A, G, I),
(B, D, F), (C, E, I), (C, D, H),
(D, G, I), (D, F, H), (E, H, I),
(F, G, H), (F, H, I)

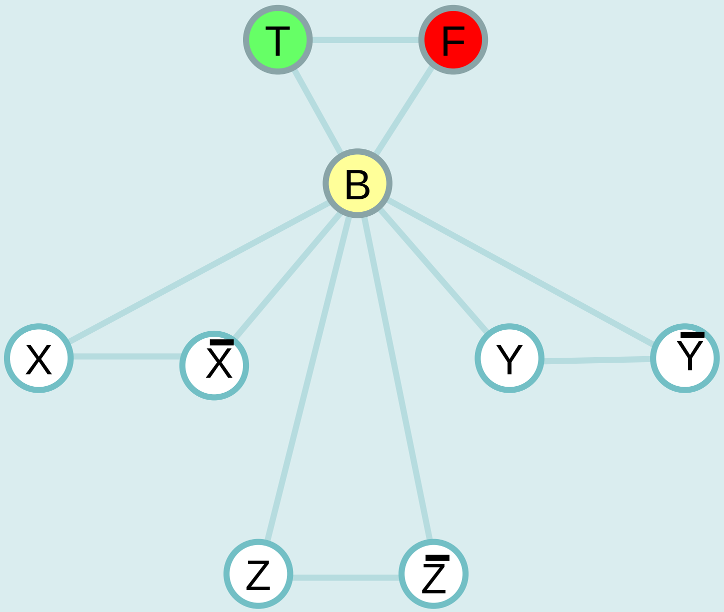
$$3DM <_P XC3$$

# Graph Coloring

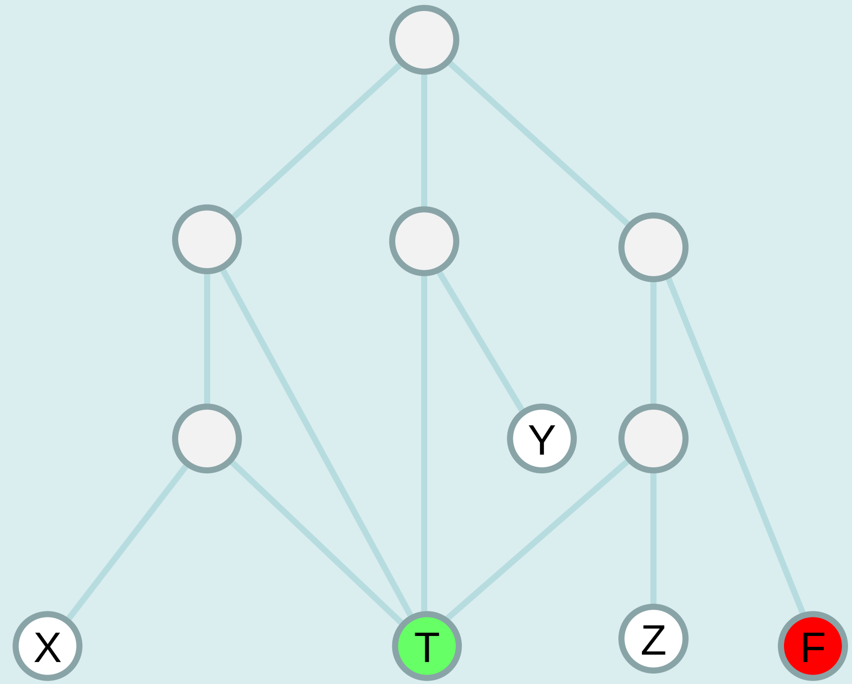
- NP-Complete
  - Graph K-coloring
  - Graph 3-coloring
- Polynomial
  - Graph 2-Coloring



# 3-SAT $<_P$ 3 Colorability




Truth Setting Gadget




Clause Testing Gadget

(Can be colored if at least one input is T)

# Number Problems

- Subset sum problem
  - Given natural numbers $w_1, . . ., w_n$ and a target number W, is there a subset that adds up to exactly W?

- Subset sum problem is NP-Complete
- Subset Sum problem can be solved in O(nW) time

# XC3 $<_P$ SUBSET SUM

Idea: Represent each set as a large integer, where the element $x_i$ is encoded as $D^i$ where D is an integer

$\{x_3, x_5, x_9\} \Rightarrow D^3 + D^5 + D^9$

Does there exist a subset that sums to exactly
$D^1 + D^2 + D^3 + \ldots + D^{n-1} + D^n$

Detail: How large is D? We need to make sure that we do not have any carries, so we can choose D = m+1, where m is the number of sets.

# Integer Linear Programming

- Linear Programming – maximize a linear function subject to linear constraints
- Integer Linear Programming – require an integer solution
- NP Completeness reduction from 3-SAT

Use 0-1 variables for $x_i$'s

Constraint for clause: $(x_1 \vee \overline{x_2} \vee \overline{x_2})$

$x_1 + (1 - x_2) + (1 - x_3) > 0$

# Scheduling with release times and deadlines (RD-Sched)

- Tasks, $\{t_1, t_2, . . . t_n\}$
- Task $t_j$ has a length $l_j$, release time $r_j$ and deadline $d_j$
- Once a task is started, it is worked on without interruption until it is completed
- Can all tasks be completed satisfying constraints?

# Subset Sum $<_P$ RD-Sched

- Subset Sum Problem
  - $\{s_1, s_2, . . . , s_N\}$, integer $K_1$
  - Does there exist a subset that sums to $K_1$?
  - Assume the total sums to $K_2$

# Reduction

- Tasks $\{t_1, t_2, \ldots t_N, x\}$
- $t_j$ has length $s_j$, release 0, deadline $K_2 + 1$
- x has length 1, release $K_1$, deadline $K_1 + 1$

# Friday: NP-Completeness and Beyond!