Computational Intractability

Let’s Review a Few Problems....

Network Design

Input: graph $G = (V, E)$ with edge costs

Minimum Spanning Tree: find minimum-cost subset of edges to connect all vertices.

$O(m \log n)$

Steiner Tree: find minimum-cost subset of edges to connect all vertices in $W \subseteq V$

No polynomial-time algorithm known!

Knapsack Problem

Input: n items with costs and weights, and capacity C

Fractional Knapsack: select fractions of each item to maximize total value without exceeding the weight capacity.

O(n log n) greedy algorithm

0-1 Knapsack: select a subset of items to maximize total value without exceeding weight capacity.

No polynomial-time algorithm known!

Tractability

- Working definition: tractable = polynomial-time
- There is a huge class of natural and interesting problems for which
  - we don't know any polynomial time algorithm
  - we can't prove that none exists

The Importance of Polynomial Time

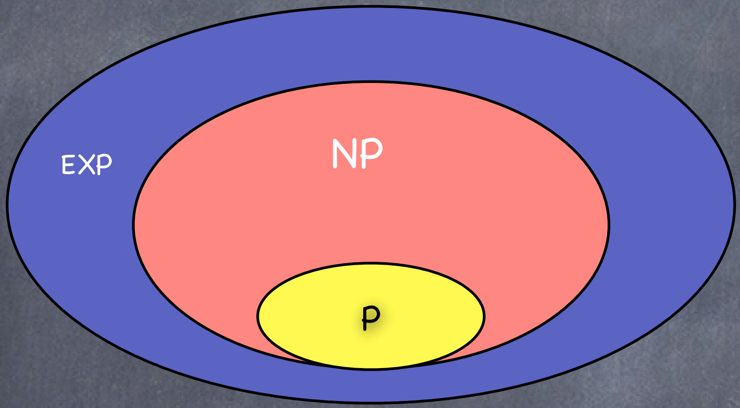
Table 2.1 The running times (rounded up) of different algorithms on inputs of increasing size, for a processor performing a million high-level instructions per second. In cases where the running time exceeds 10^{25} years, we simply record the algorithm as taking a very long time.

	n	$n \log_2 n$	n^2	n^3	1.5^n	2^n	$n!$
$n = 10$	< 1 sec	< 1 sec	< 1 sec	< 1 sec	< 1 sec	< 1 sec	4 sec
$n = 30$	< 1 sec	< 1 sec	< 1 sec	< 1 sec	< 1 sec	18 min	10^{25} years
$n = 50$	< 1 sec	< 1 sec	< 1 sec	< 1 sec	11 min	36 years	very long
$n = 100$	< 1 sec	< 1 sec	< 1 sec	1 sec	12,892 years	10^{17} years	very long
$n = 1{,}000$	< 1 sec	< 1 sec	1 sec	18 min	very long	very long	very long
$n = 10{,}000$	< 1 sec	< 1 sec	2 min	12 days	very long	very long	very long
$n = 100{,}000$	< 1 sec	2 sec	3 hours	32 years	very long	very long	very long
$n = 1{,}000{,}000$	1 sec	20 sec	12 days	31,710 years	very long	very long	very long

Polynomial

Not polynomial

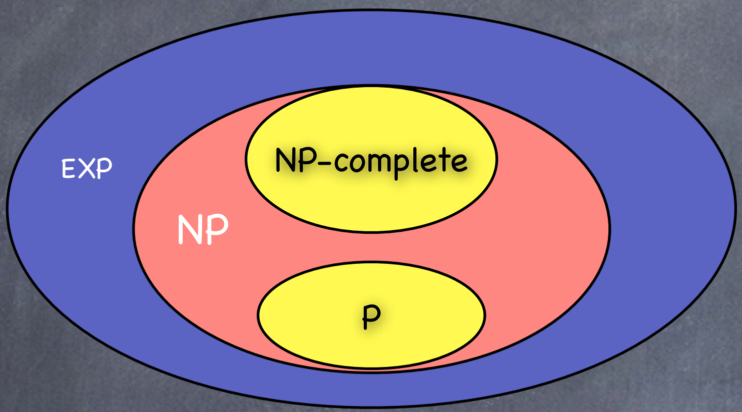
Preview of Landscape: Known Classes of Problems



P: polynomial time
NP: class that includes most
most of the problems
we don't know about
EXP: exponential time

Goal 1: characterize problems we don't know about by defining the class NP

NP-completeness



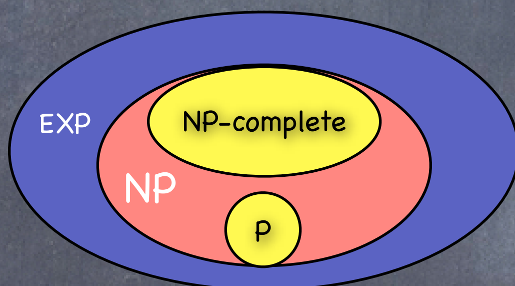
NP-complete: class of problems that are "as hard" as every other problem in NP

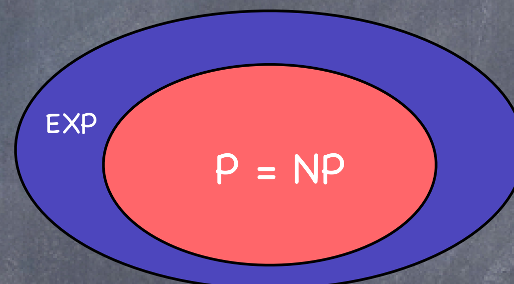
A polynomial-time algorithm for any NP-complete problem implies one for every problem in NP

Goal 2: understand NP-completeness

P != NP?

Two possibilities (we don't know which is true, but we think P != NP)





$1M prize if you can figure out the answer
(one of Clay institute's seven Millennium Problems)

Goals

Develop tools to classify problems within this landscape and understand the implications

- Polynomial Time Reductions: make statements about relative hardness of problems
- NP: characterize the class of problems that includes both P and most known "hard" problems
- NP-completeness: show that certain problems are as hard as any others in NP

Polynomial Time Reductions

Reduction

- Map problem Y to a different problem X that we know how to solve
- Solve problem X
- Mapping solution of X back to a solution of Y

- We've seen many reductions already

Reduction Example

Problem Y: given flight segments and maintenance time, determine how to schedule airplanes to cover all flight segments

1. Map to different problem X that we know how to solve (X = network flow):
   - Nodes are city/time combinations
   - Edges are flight segments
   - Etc..

Reduction Example

2. Solve problem X (use Ford-Fulkerson)
3. Map solution of X back to solution of Y
   - Assign a different airplane to each s-t path with flow = 1

Polynomial-Time Reduction

- Reduction. Problem Y is polynomial-time reducible to problem X if arbitrary instances of problem Y can be solved using:
  - Polynomial number of standard computational steps, plus
  - Polynomial number of calls to black-box that solves problem X

- Notation. $Y \leq_P X$.

- Conclusion. If X can be solved in polynomial time and $Y \leq_P X$, then Y can be solved in polynomial time.

Polynomial-Time Reduction

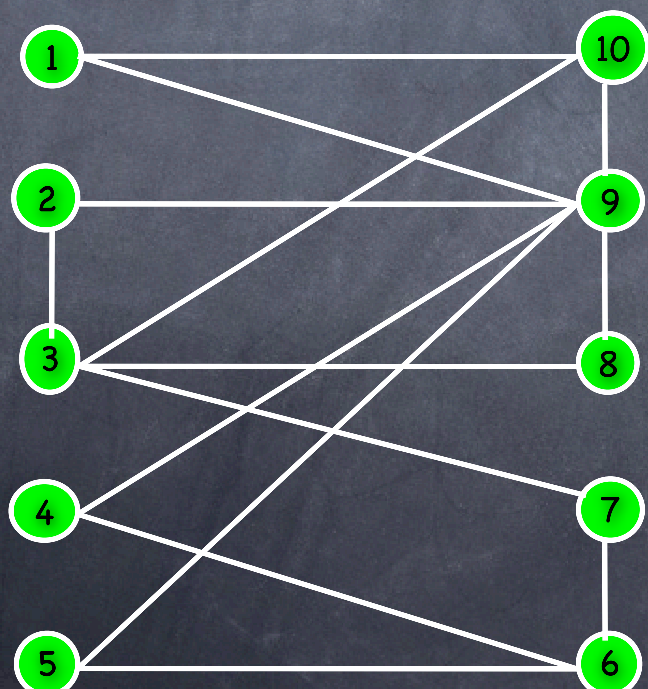
- Classify problems according to relative difficulty.
- Consequences of $Y \leq_P X$
  - New algorithms. If X can be solved in polynomial-time, then Y can also be solved in polynomial time.
  - Intractability. If Y cannot be solved in polynomial-time, then X cannot be solved in polynomial time.

Basic Reduction Strategies

- Reduction by simple equivalence.
- Reduction from special case to general case.
- Reduction by encoding with gadgets.

Independent Set

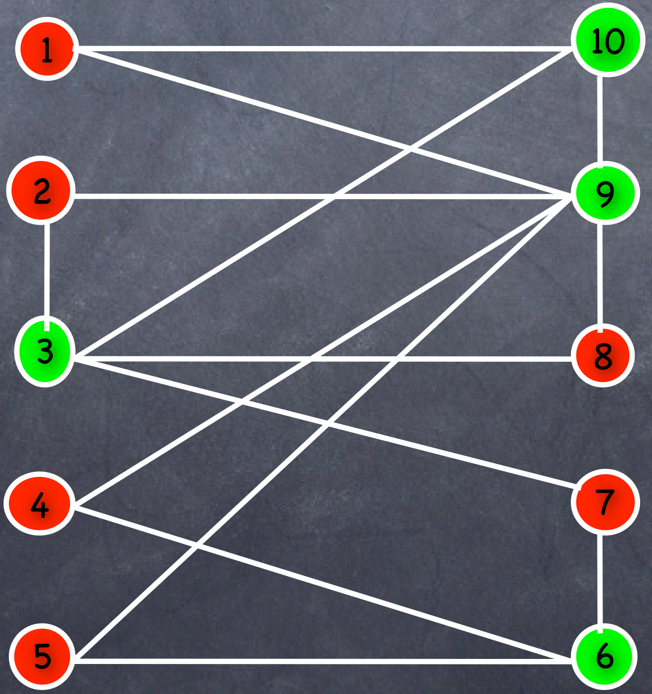
- INDEPENDENT SET: Given a graph $G = (V, E)$ and an integer k, is there a subset of vertices $S \subseteq V$ such that $|S| \geq k$, and for each edge at most one of its endpoints is in S?



What is the largest independent set?

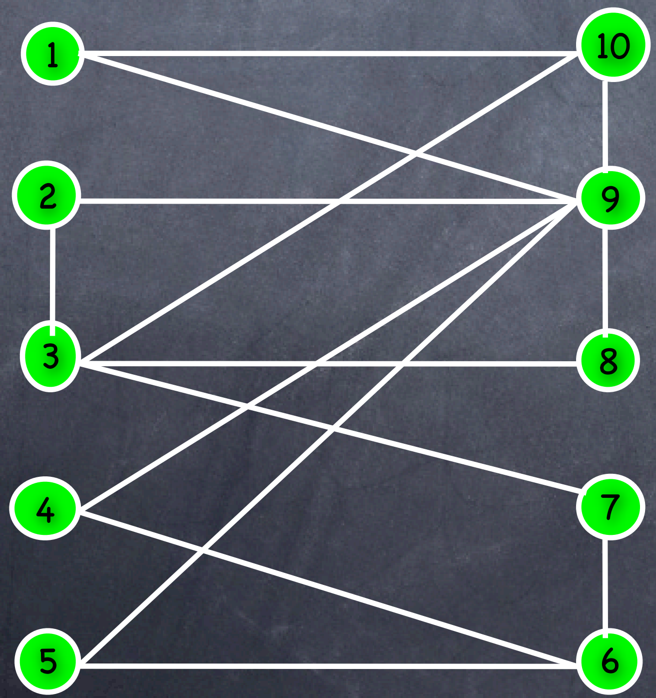
Independent Set

- INDEPENDENT SET: Given a graph $G = (V, E)$ and an integer k, is there a subset of vertices $S \subseteq V$ such that $|S| \geq k$, and for each edge at most one of its endpoints is in S?



Vertex Cover

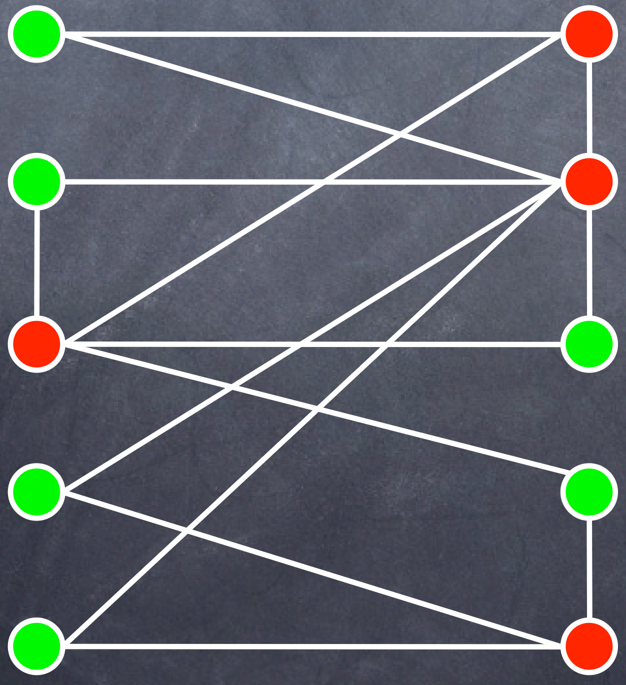
- VERTEX COVER: Given a graph G = (V, E) and an integer k, is there a subset of vertices $S \subseteq V$ such that $|S| \leq k$, and for each edge, at least one of its endpoints is in S?



What is the smallest vertex cover?

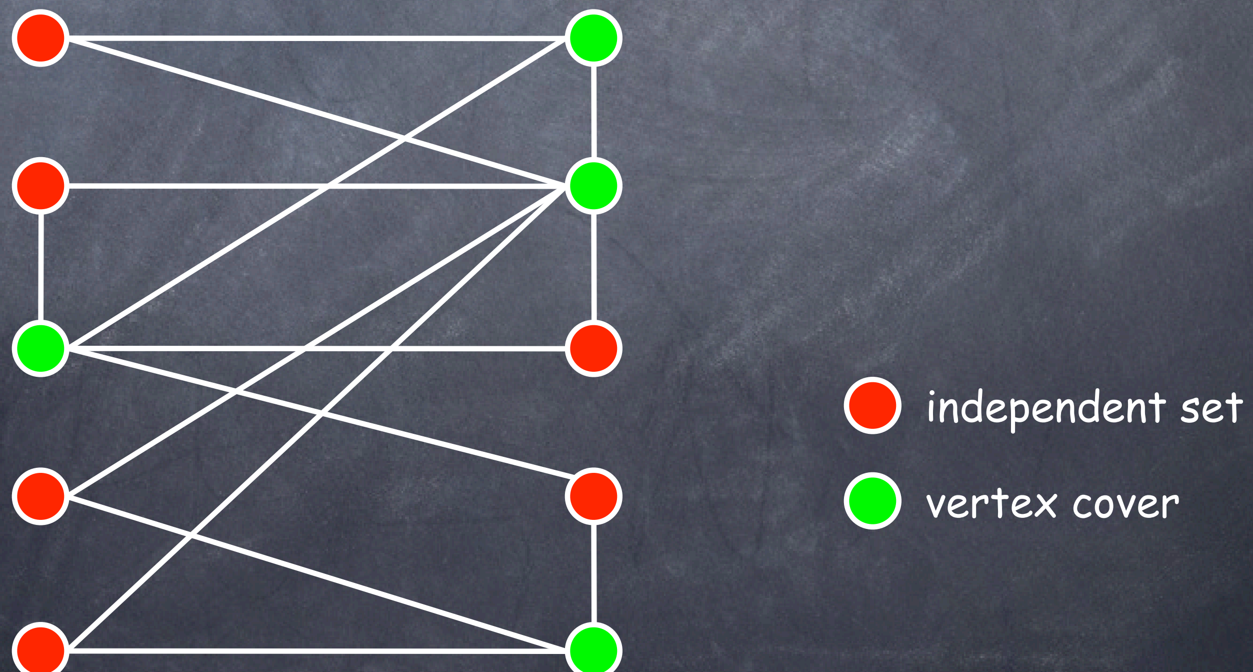
Vertex Cover

- VERTEX COVER: Given a graph $G = (V, E)$ and an integer k, is there a subset of vertices $S \subseteq V$ such that $|S| \leq k$, and for each edge, at least one of its endpoints is in S?



Vertex Cover and Independent Set

- Claim. S is an independent set iff $V - S$ is a vertex cover.



Vertex Cover and Independent Set

- Claim. S is an independent set iff $V - S$ is a vertex cover.

- Proof of if-part:
  - Let S be any independent set.
  - Consider an arbitrary edge (u, v).
  - S independent $\Rightarrow u \notin S$ or $v \notin S$
    $\Rightarrow u \in V - S$ or $v \in V - S$.
  - Thus, $V - S$ covers (u, v).

- Proof of only-if-part: similar

Vertex Cover and Independent Set

- Claim. VERTEX-COVER $\leq_p$ INDEPENDENT-SET
- Proof. Given graph G = (V, E) and integer k, return "yes" iff G has an independent set of size at least n-k.
  (Is this polynomial?)

- Claim. INDEPENDENT-SET $\leq_p$ VERTEX-COVER
- Proof. similar

Basic Reduction Strategies

- Reduction by simple equivalence.
- Reduction from special case to general case.
- Reduction by encoding with gadgets.

Set Cover Problem

- You want all towns in the county to be within 15 minutes driving time of some fire station.
- Goal: build as few fire stations as possible satisfying the driving time constraint.
- (Station covers set of towns)

Set Cover

	Amherst	Granby	Hadley	Pelham	South Hadley
Amherst	0	20	8	17	19
Granby	20	0	21	23	9
Hadley	8	21	0	25	15
Pelham	17	23	25	0	31
South Hadley	19	9	15	31	0

Set Cover

- SET COVER: Given a set U of elements, a collection S_1, S_2, . . . , S_m of subsets of U, and an integer k, does there exist a collection of $\leq$ k of these sets whose union is equal to U?

U = {A, G, H, P, SH}

S1 = {A, H}
S2 = {G, SH}
S3 = {A, H, SH}

S4 = {P}
S5 = {G, H, SH}

Set Cover

- SET COVER: Given a set U of elements, a collection S_1, S_2, . . . , S_m of subsets of U, and an integer k, does there exist a collection of ≤ k of these sets whose union is equal to U?

U = {A, G, H, P, SH}

S1 = {A, H}
S2 = {G, SH}
S3 = {A, H, SH}

S4 = {P}
S5 = {G, H, SH}

k = 3

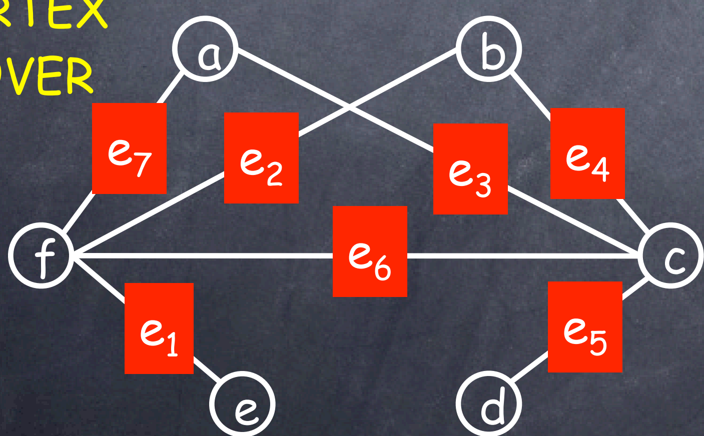
Vertex Cover is Reducible to Set Cover

- Claim. VERTEX-COVER $\leq_P$ SET-COVER.
- Proof. Given a VERTEX-COVER instance $G = (V, E)$ and k, we construct a set cover instance whose size equals the size of the vertex cover instance.
- Exercise

Vertex Cover is Reducible to Set Cover

- Step 1: Map the vertex cover problem into a set cover problem
  - U is the set of all edges
  - For each vertex v, create a set $S_v = \{e \in E : e \text{ incident to } v\}$

VERTEX COVER



SET COVER

$U = \{1, 2, 3, 4, 5, 6, 7\}$

$S_a = \{3, 7\}$ $S_b = \{2, 4\}$

$S_c = \{3, 4, 5, 6\}$ $S_d = \{5\}$

$S_e = \{1\}$ $S_f = \{1, 2, 6, 7\}$

Vertex Cover is Reducible to Set Cover

- Step 2: Solve the Set Cover problem using the same value for k:
  - Is there a collection of at most k sets such that their union is U?

Solving for k = 2

SET COVER

$U = \{ 1, 2, 3, 4, 5, 6, 7 \}$

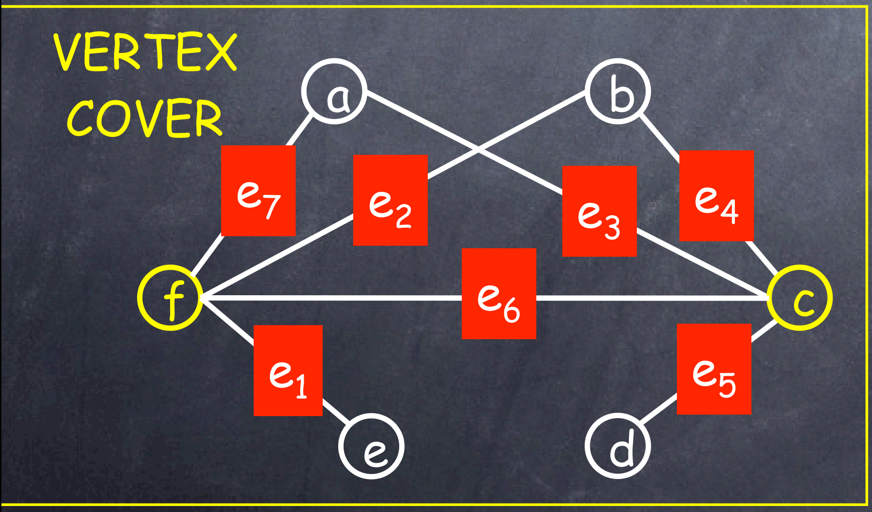
$S_a = \{3, 7\}$ $S_b = \{2, 4\}$

$S_c = \{3, 4, 5, 6\}$ $S_d = \{5\}$

$S_e = \{1\}$ $S_f = \{1, 2, 6, 7\}$

Vertex Cover is Reducible to Set Cover

- Step 3: Map the set cover solution back to a vertex cover solution
  - For each set in the set cover solution, select the corresponding vertex in the vertex cover problem



SET COVER

$U = \{1, 2, 3, 4, 5, 6, 7\}$

$S_a = \{3, 7\}$ $S_b = \{2, 4\}$

$S_c = \{3, 4, 5, 6\}$ $S_d = \{5\}$

$S_e = \{1\}$ $S_f = \{1, 2, 6, 7\}$

Basic Reduction Strategies

- Reduction by simple equivalence.
- Reduction from special case to general case.
- Reduction by encoding with gadgets.

Satisfiability

- **Term:** A Boolean variable or its negation.
  x_i OR $\overline{x_i}$
- **Clause:** A disjunction ("or") of terms.
  $C_j = x_1 \vee x_2 \vee x_3$
- **Formula Φ:** A conjunction ("and") of clauses
  $C_1 \wedge C_2 \wedge C_3 \wedge C_4$
- **SAT:** Given a formula, is there a truth assignment that satisfies all clauses? (i.e. all clauses evaluate to "true")
- **3-SAT:** SAT where each clause contains exactly 3 terms
  $(x_1 \vee x_2 \vee x_3) \wedge (x_1 \vee x_2 \vee x_3) \wedge (x_1 \vee x_2 \vee x_3)$

3-SAT is Reducible to Independent Set

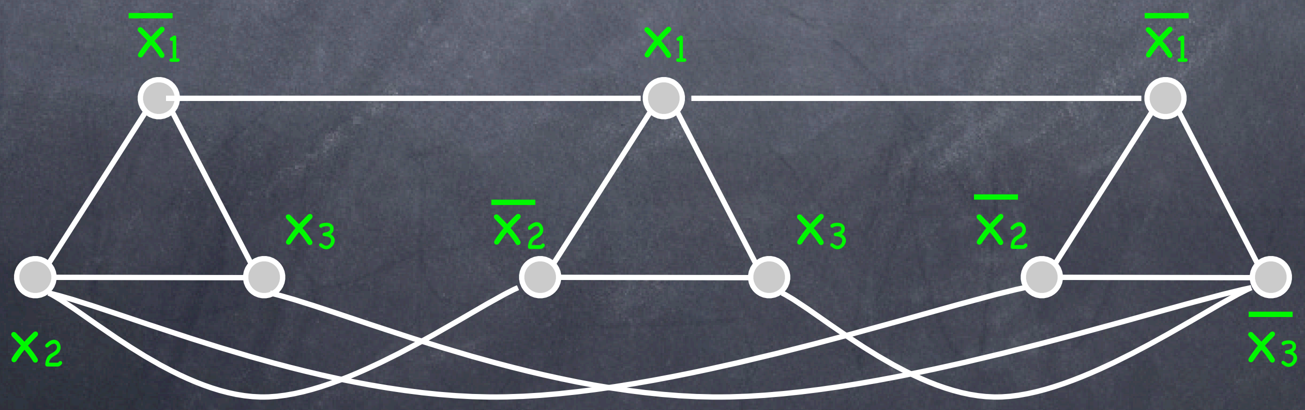
- Claim. 3-SAT $\leq_P$ INDEPENDENT-SET.
- Proof. Given an instance Φ of 3-SAT, we construct an instance (G, k) of INDEPENDENT-SET that has an independent set of size k iff Φ is satisfiable.

3 Satisfiability Reduces to Independent Set

- Claim. 3-SAT $\leq_P$ INDEPENDENT-SET.

- Construction.
  - G contains 3 vertices for each clause, one for each term.
  - Connect 3 terms in a clause in a triangle.
  - Connect term to each of its negations.

G

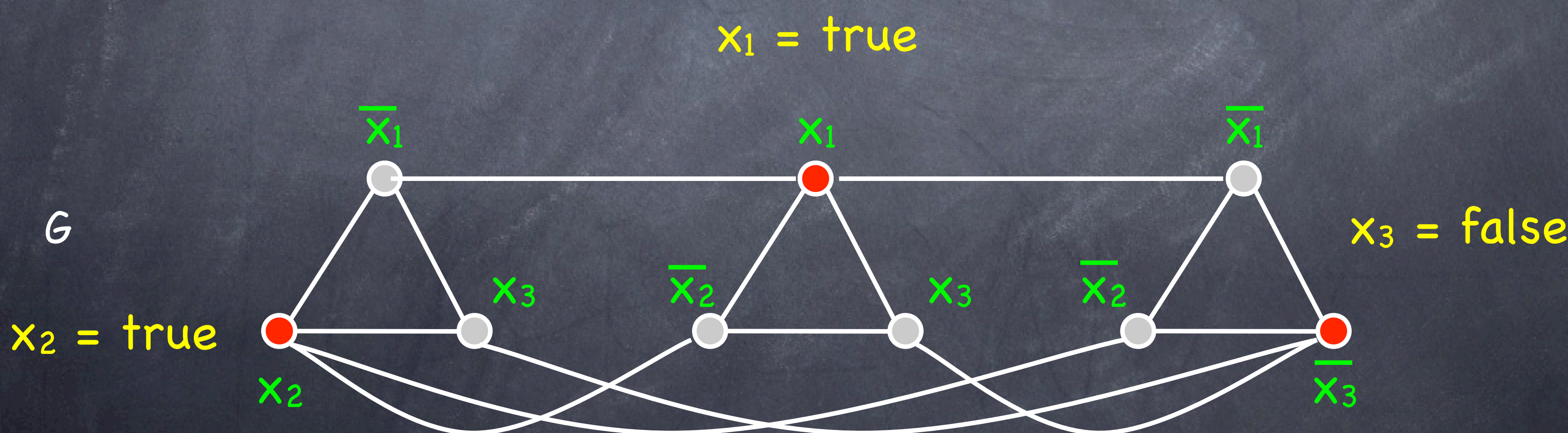


k = 3

$(\overline{x_1} \lor x_2 \lor x_3) \land (x_1 \lor \overline{x_2} \lor x_3) \land (\overline{x_1} \lor \overline{x_2} \lor \overline{x_3})$

3 Satisfiability Reduces to Independent Set

- **Claim.** 3-SAT $\leq_P$ INDEPENDENT-SET.
- With an independent set solution, we can derive a SAT assignment.



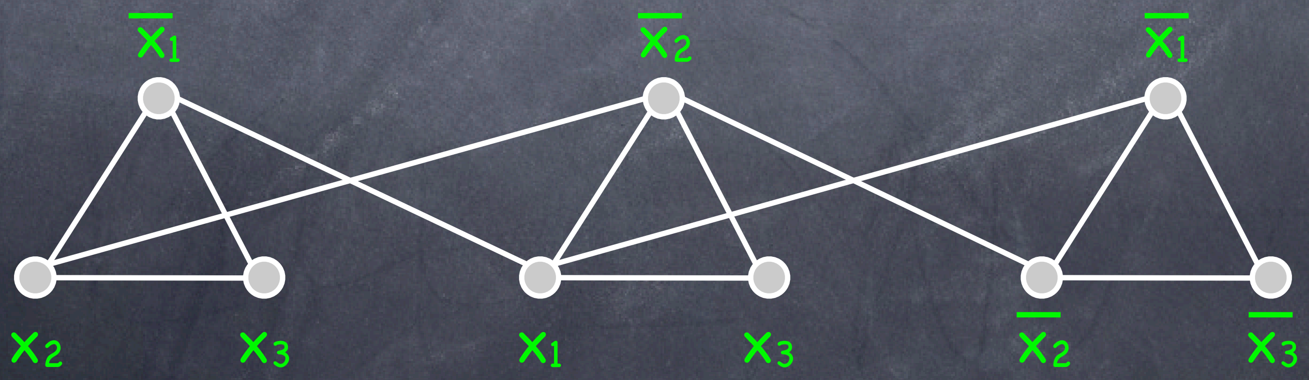
k = 3

$(\overline{x_1} \vee x_2 \vee x_3) \wedge (x_1 \vee \overline{x_2} \vee x_3) \wedge (\overline{x_1} \vee \overline{x_2} \vee \overline{x_3})$

3 Satisfiability Reduces to Independent Set

- Claim. G contains independent set of size $k = |\Phi|$ iff Φ is satisfiable.
- Proof of if-part: Let S be independent set of size k.
  - S must contain exactly one vertex in each triangle.
  - Set these terms to true.
  - Truth assignment is consistent and all clauses are satisfied.

G



k = 3

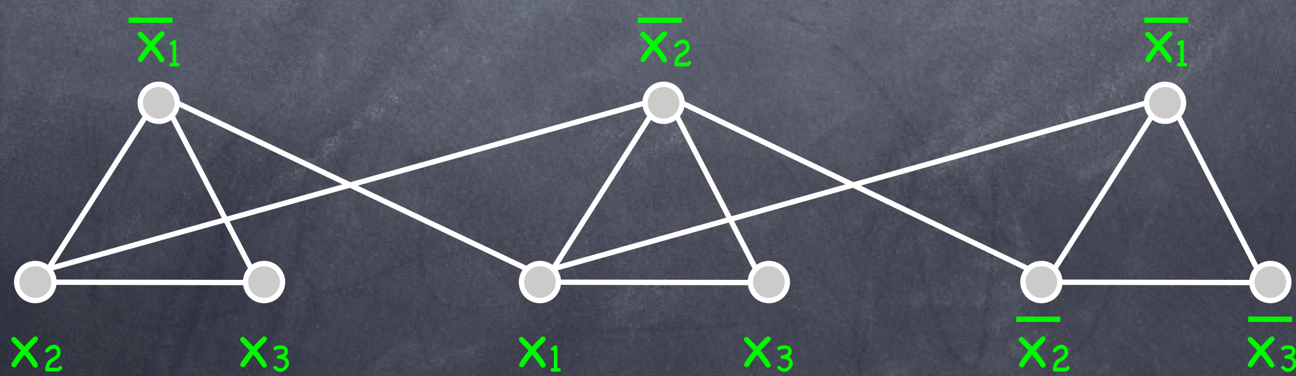
$$(\overline{x_1} \vee x_2 \vee x_3) \wedge (x_1 \vee \overline{x_2} \vee x_3) \wedge (\overline{x_1} \vee \overline{x_2} \vee \overline{x_3})$$

3 Satisfiability Reduces to Independent Set

- **Claim.** G contains independent set of size $k = |\Phi|$ iff Φ is satisfiable.

- **Proof of only-if part:** Given satisfying assignment, select one true term from each triangle. This is an independent set of size k. ▪

G



$k = 3$

$$(\overline{x_1} \vee x_2 \vee x_3) \wedge (x_1 \vee \overline{x_2} \vee x_3) \wedge (\overline{x_1} \vee \overline{x_2} \vee \overline{x_3})$$

Review

- Basic reduction strategies.
  - Simple equivalence: INDEPENDENT-SET $\equiv_p$ VERTEX-COVER.
  - Special case to general case: VERTEX-COVER $\leq_p$ SET-COVER.
  - Encoding with gadgets: 3-SAT $\leq_p$ INDEPENDENT-SET.
- Transitivity. If $X \leq_p Y$ and $Y \leq_p Z$, then $X \leq_p Z$.
- Proof idea: Compose the two algorithms.
- Example: 3-SAT $\leq_p$ INDEPENDENT-SET $\leq_p$ VERTEX-COVER $\leq_p$ SET-COVER.