CS 580
ALGORITHM DESIGN AND ANALYSIS

NP and Computational Tractability

Vassilis Zikas

SUMMARY

- So far:
  - Algorithms
- For the next few lectures:
  - Lack of algorithms!

PATTERNS

- Algorithms so far had “design patterns”
  - Greedy
  - D&C
  - Dynamic Programming
  - Duality
  - Local search
  - Randomized algorithms
- We’ve been amazingly successful!
  - E.g. a graph with n vertices could have n^{n-2} spanning trees, but we found the minimum weight one in time $O(|E| + nlogn)$
- However, this is not really representative of reality
  - For most problems out there, we have no idea how to navigate among the exponentially many possible solutions

REDUCTIONS

- Since we can’t solve these problems, why don’t we try to at least classify them?
  - Proposed category 1: Those that we can solve in polynomial time
  - Proposed category 2: Those that we cannot solve in polynomial time
- Frustrating news: Huge number of fundamental problems have defied classification for years
- This module: Show that these problems are “computationally equivalent”
  - Different manifestations of a single hard problem

REDUCTIONS

- Suppose we could solve a problem X in polynomial time
- What else could we do in polynomial time?
- We've already seen this: circulation via network flow
- Reductions: A problem X is polynomial time reducible to a problem Y if arbitrary instances of problem X can be solved using:
  - Polynomial number of "standard" computational steps
  - Polynomial number of calls to an "oracle" that solves problem Y
- We will also say "X reduces to Y" or "Y is at least as hard as X"
- Notation: $X \leq_P Y$

REDUCTIONS

- Note: We do pay for writing down instances of Y, so the instances of Y we ask our oracle to solve should have polynomial size (in terms of the size of the input of X)
- This notion of reduction is called a Cook reduction (named after Stephen Cook)
  - Different than Karp reductions

REDUCTIONS

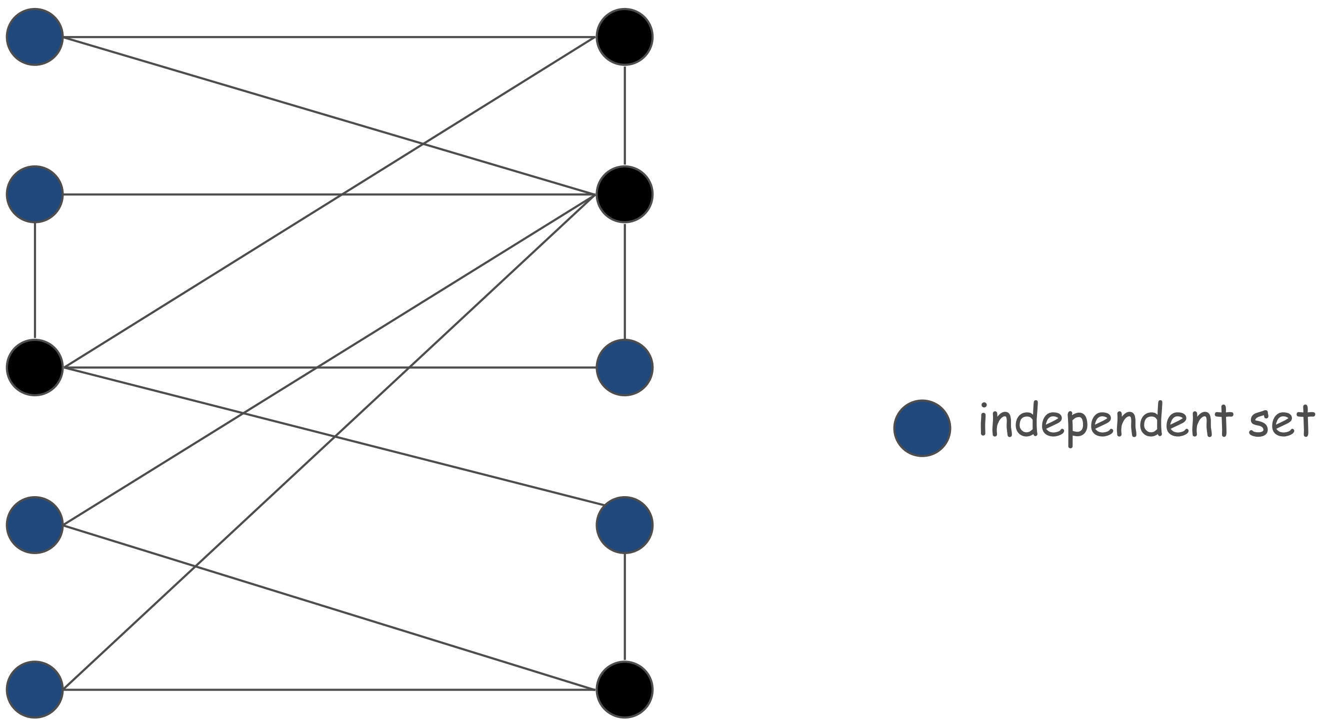
- Purpose: Classify problems according to relative difficulty
- Positive results: If $X \leq_P Y$ and Y can be solved in polynomial time, then X can be solved in polynomial time
- Negative results (intractability): If $X \leq_P Y$ and X cannot be solved in polynomial time, then Y cannot be solved in polynomial time
- Equivalence: If $X \leq_P Y$ and $Y \leq_P X$, then we use notation $X \equiv_P Y$
- Absolutely amazing…

A FIRST REDUCTION

- Independent set and Vertex Cover

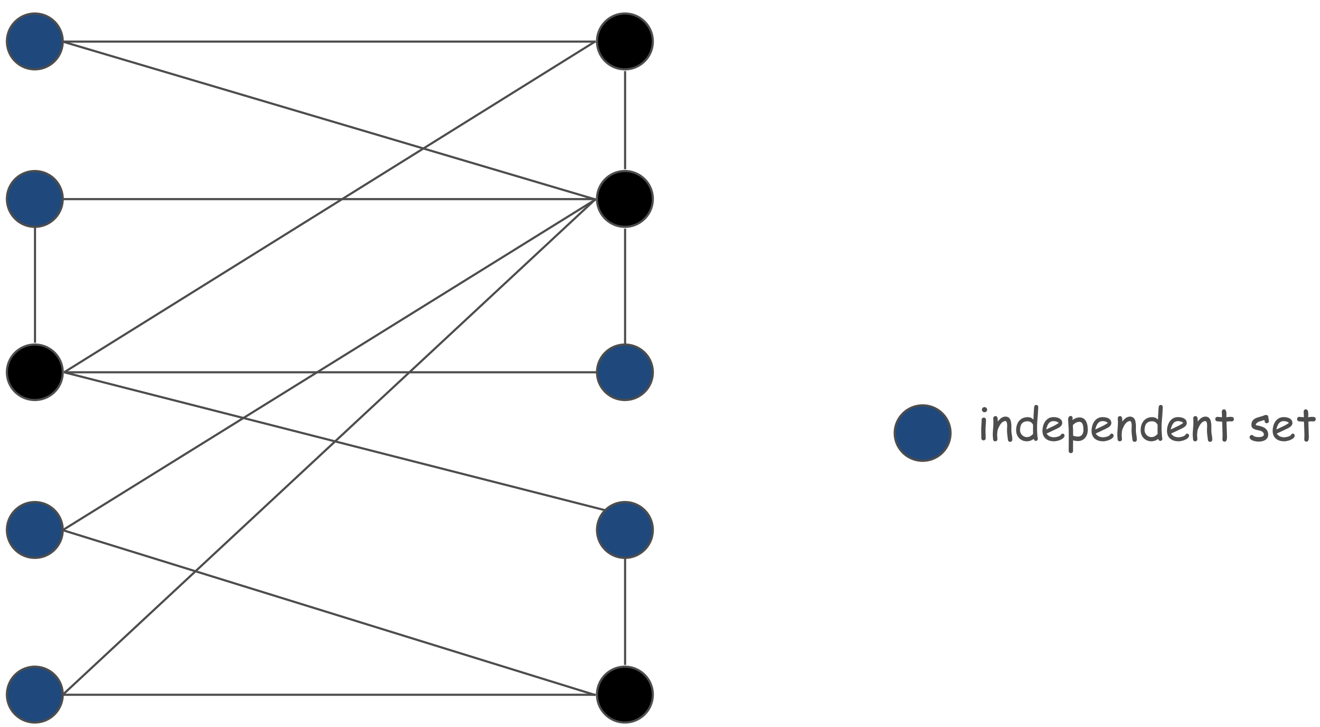
INDEPENDENT SET

- Input: Undirected graph $G = (V, E)$ and an integer k
- Question: Is there an independent set of size at least k?
    - $S \subseteq V$ is an independent set if for every edge $e \in E$, at most one of its endpoints in S



INDEPENDENT SET

- Is there an IS of size at least 6?
  - Yes
- Is there an IS of size at least 7?
  - No

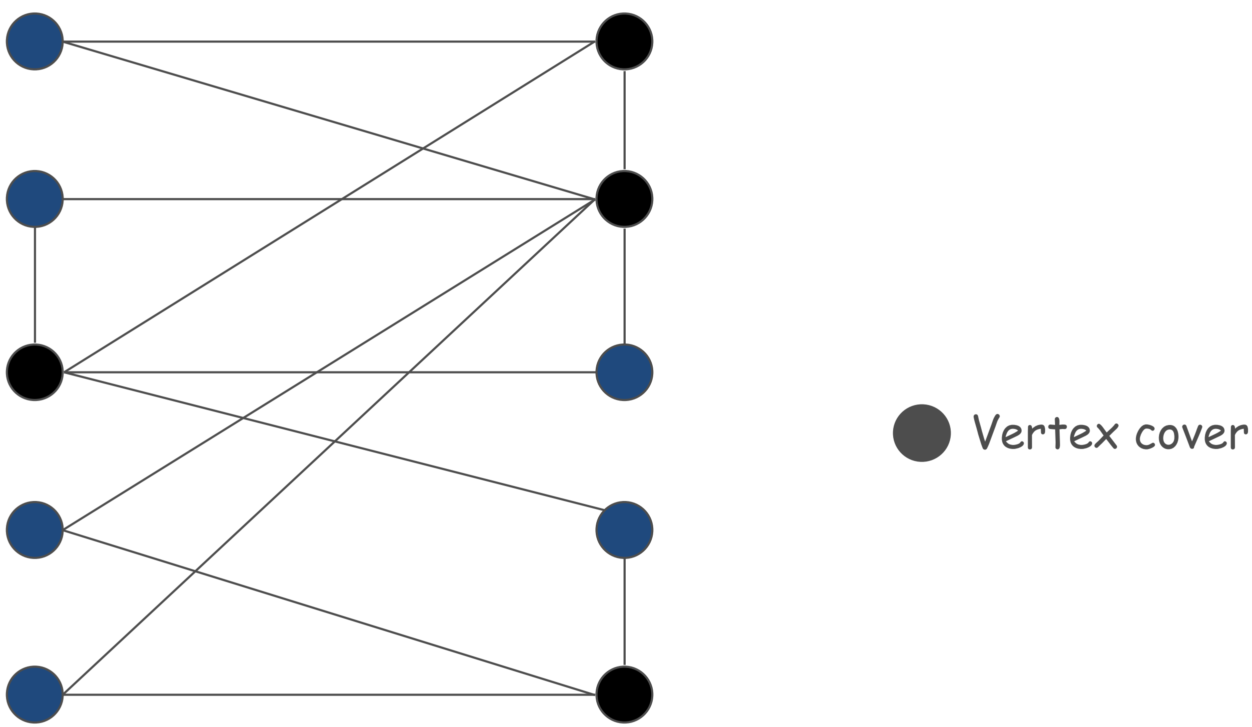


VERTEX COVER

- Input: Undirected graph $G = (V, E)$ and an integer k
- Question: Is there a vertex cover of size at most k?
  - $S \subseteq V$ is a vertex cover if for every edge $e \in \mathrm{E}$, at least one of its endpoints is in S

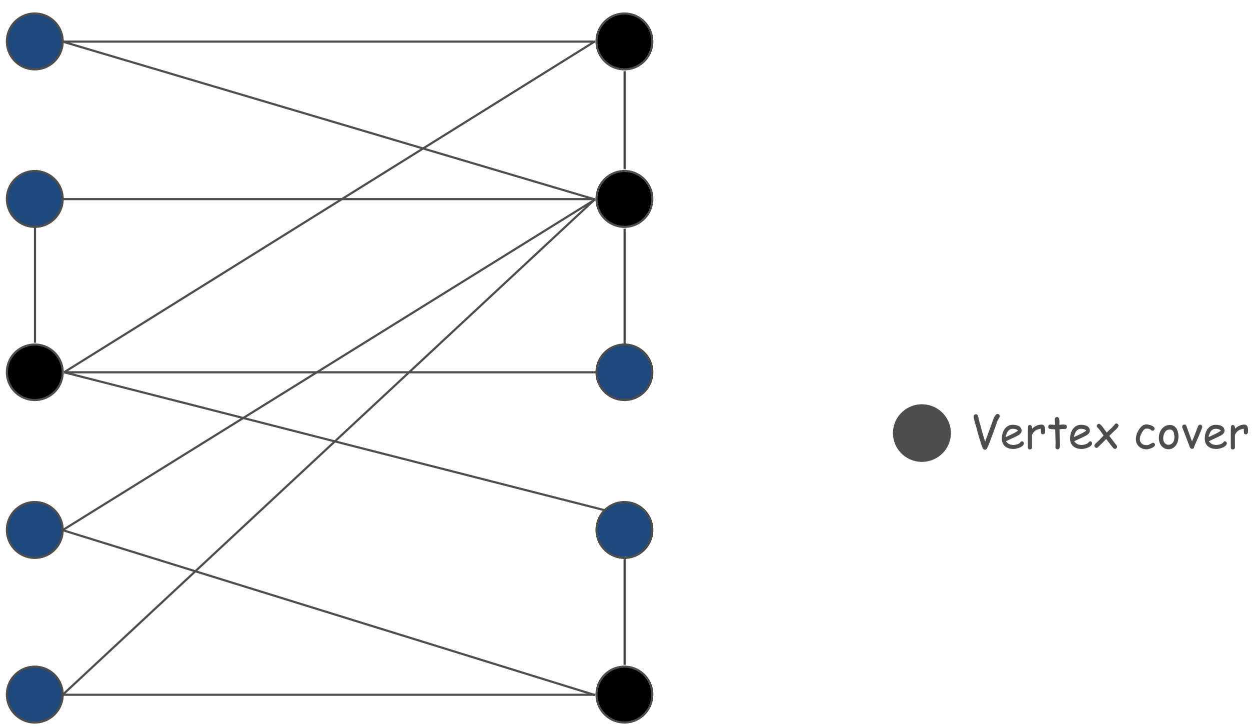
VERTEX COVER

- Input: Undirected graph $G = (V, E)$ and an integer k
- Question: Is there a vertex cover of size at most k?
  - $S \subseteq V$ is a vertex cover if for every edge $e \in$ E, at least one of its endpoints is in S



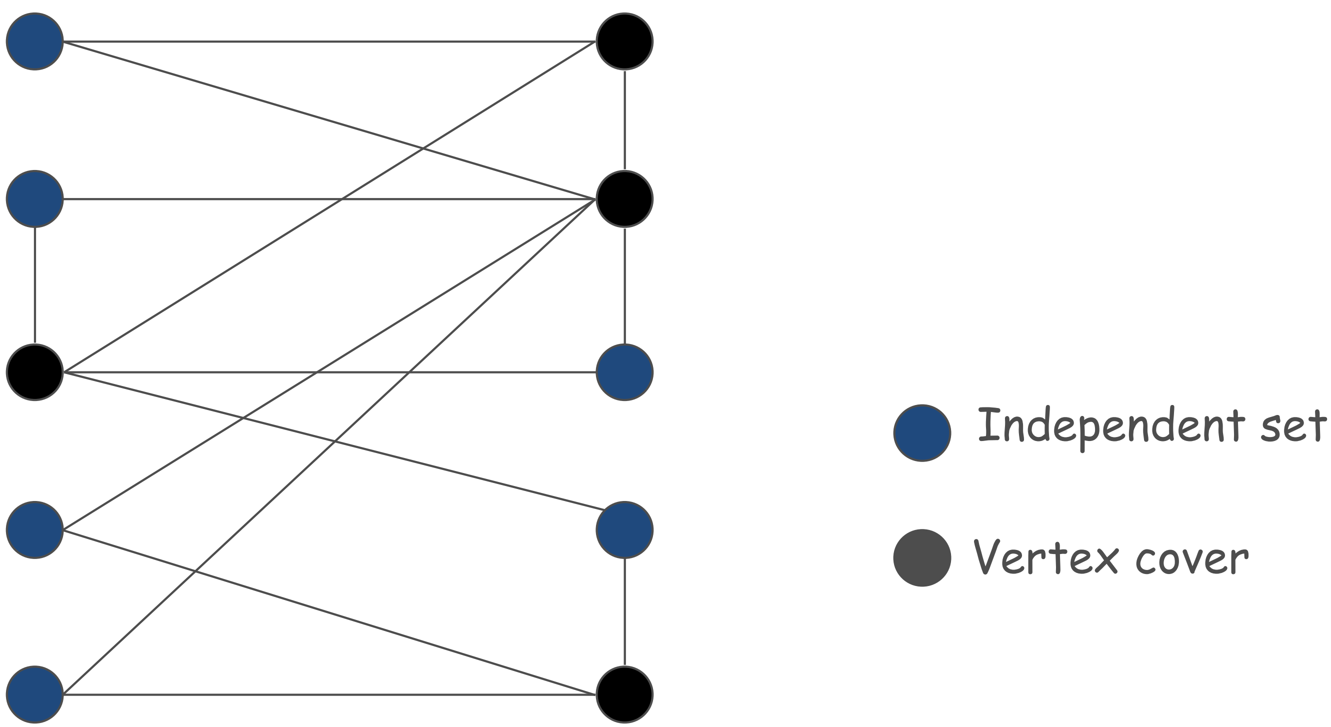
VERTEX COVER

- Is there a vertex cover of size at most 4?
  - Yes
- Is there a vertex cover of size at most 3?
  - No



A FIRST REDUCTION

- Claim: INDEPENDENT SET $\equiv_P$ VERTEX COVER
- Proof: We'll show that if S is an independent set, then $V \setminus S$ is a vertex cover



A FIRST REDUCTION

- Claim: INDEPENDENT SET $\equiv_P$ VERTEX COVER
- (1)
  - Let S be any independent set
  - Consider an arbitrary edge (u, v)
  - At least one of the following holds: $u \notin S$ or v $\notin S$
  - Thus, $u \in V \setminus S$ or $v \in V \setminus S$
  - Thus, $V \setminus S$ covers (u, v)
- (2)
  - Let $V \setminus S$ be any vertex cover
  - Consider two nodes $v \in S$ and $u \in S$
  - Since $V \setminus S$ is a vertex cover, $(u, v) \notin E$
  - Thus, no two nodes in S have an edge between them
  - Thus, S is an independent set

A FIRST REDUCTION

- Claim: INDEPENDENT SET $\equiv_P$ VERTEX COVER
- Proof:
  - INDEPENDENT SET $\leq_P$ VERTEX COVER:
    - Say we have an oracle for VERTEX COVER
    - If you want to know where there exists an IS with size at least k ask the oracle if there exists a VC with size at most $n - k$
  - INDEPENDENT SET $\geq_P$ VERTEX COVER:
    - Say we have an oracle for INDEPENDENT SET
    - If you want to know where there exists a VC with size at most k ask the oracle if there exists an IS with size at least $n - k$.

A SECOND REDUCTION

- Simple reduction strategy:
  - From special case to general case
  - E.g. if a problem is difficult for bipartite graphs it should be difficult for general graphs

SET COVER

- Input: A set U of elements, a collection $S_1, S_2, \ldots, S_m$ of subsets of U and an integer k
- Question: Does there exist a collection of at most k of these subsets whose union equals U?
  - $\exists S_{i_1}, S_{i_2}, \ldots, S_{i_k}: \cup_{j=1}^{k} S_{i_j} = U$?

SET COVER

- Example:
  - $U = \{1, 2, 3, 4, 5, 6, 7\}$
  - $k = 2$
  - $S_1 = \{3, 7\}$
  - $S_2 = \{3, 4, 5, 6\}$
  - $S_3 = \{1\}$

$S_4 = \{2, 4\}$

$S_5 = \{5\}$

$S_6 = \{1, 2, 6, 7\}$

SET COVER

- Example:
  - $U = \{1, 2, 3, 4, 5, 6, 7\}$
  - $k = 2$
  - $S_1 = \{3, 7\}$
  - $S_2 = \{3, 4, 5, 6\}$
  - $S_3 = \{1\}$

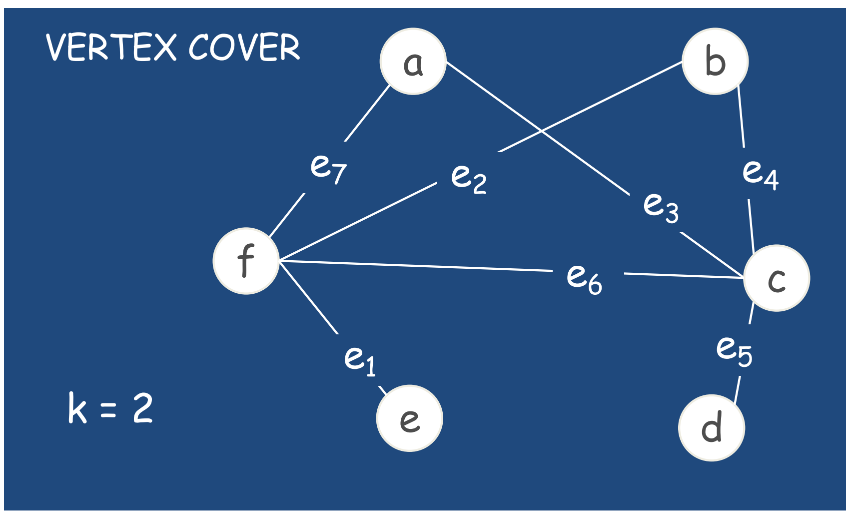
$S_4 = \{2, 4\}$

$S_5 = \{5\}$

$S_6 = \{1, 2, 6, 7\}$

SET COVER AND VERTEX COVER

- Claim: VERTEX COVER $\leq_P$ SET COVER
- Proof: Given an instance of VERTEX COVER, i.e. a graph $G = (V, E)$ and an integer k, we construct a set cover instance with size at most the size of the vertex cover
- Construction:
  - $k = k, U = E, S_v = \{e \in E : e \text{ incident to } v\}$
  - Set cover size $\leq k$ if and only if vertex cover $\leq k$



SET COVER

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

$S_a = \{3, 7\}$ $\quad$ $S_b = \{2, 4\}$
$S_c = \{3, 4, 5, 6\}$ $\quad$ $S_d = \{5\}$
$S_e = \{1\}$ $\quad$ $S_f = \{1, 2, 6, 7\}$

A THIRD REDUCTION

- Gadgets!
  - Most common reduction “type”

SATISFIABILITY

- Literal: A Boolean variable, or its negation
  - x_i or $\overline{x_i}$
- Clause: A disjunction (OR) of literals
  - $C_j = x_1 \vee x_5 \vee \overline{x_6} \vee x_{11}$
- Conjuctive normal form: A propositional formula ϕ that is the conjuction of clauses
  - $\phi = C_1 \wedge C_2 \wedge C_3 \wedge C_4$
- **SAT** problem: Given a CNF formula ϕ, decide whether there exists a satisfying assignment
  - That is, an assignment $x_1 = true, x_2 = false, \ldots,$ such that ϕ is $true$
- **3-SAT:** Every clause has at most 3 literals
  - **E.g.** $(x_1 \vee x_4 \vee \overline{x_2}) \wedge (\overline{x_1} \vee x_2 \vee \overline{x_3}) \wedge (x_1 \vee x_3)$
  - **Yes:** $x_1 = 1, x_2 = 1, x_3 = 1, x_4 = 0$

SATISFIABILITY

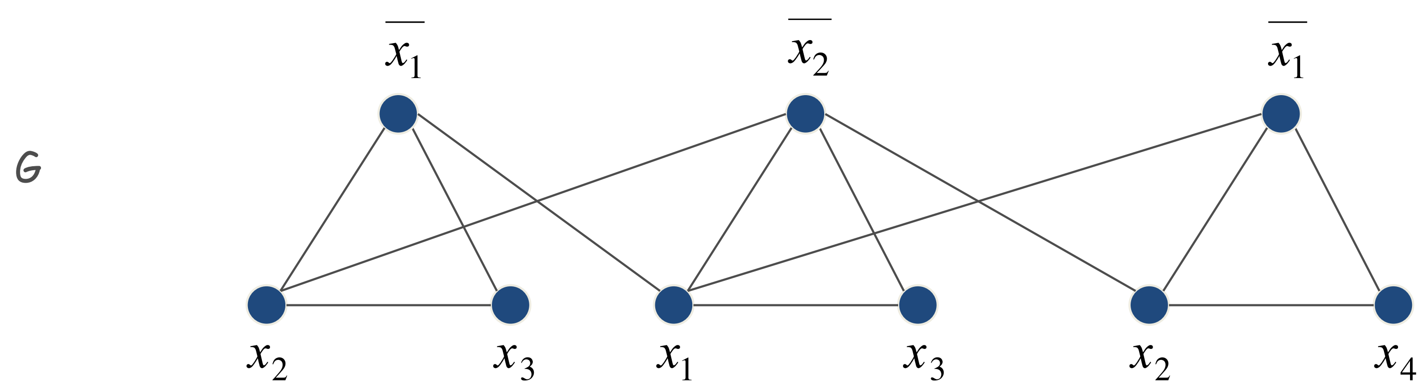
- Claim: 3-SAT $\leq_P$ INDEPENDENT SET
- Thinking about 3-SAT
  - Make a 0/1 assignment to each of the variables and manage to satisfy every clause (at least one of the 3 literals in every clause)
  - Select a literal from each clause to satisfy the clause. Selection across clauses has no conflicts
    - E.g. not using x_1 to satisfy C_1 and $\overline{x_1}$ to satisfy C_2

SATISFIABILITY

- Claim: 3-SAT $\leq_P$ INDEPENDENT SET
- Proof: Given a CNF formula ϕ with k clauses
- Construction:
  - G contains 3 vertices for each clause, one for each literal
  - Connect 3 literals in a clause in a triangle
  - Connect literal to its negations (across clauses)

SATISFIABILITY

- Claim: 3-SAT $\leq_P$ INDEPENDENT SET
- Proof: Given a CNF formula ϕ with k clauses
- Construction:
  - G contains 3 vertices for each clause, one for each literal
  - Connect 3 literals in a clause in a triangle
  - Connect literal to its negations (across clauses)



k = 3 $\quad \Phi = \left(\overline{x_1} \vee x_2 \vee x_3\right) \wedge \left(x_1 \vee \overline{x_2} \vee x_3\right) \wedge \left(\overline{x_1} \vee x_2 \vee x_4\right)$

SATISFIABILITY

- Claim: 3-SAT $\leq_P$ INDEPENDENT SET
- Proof: Let S be an independent set of size k
  - S must contain exactly one vertex from each triangle
  - Set the corresponding literals to $true$ (and all other variables consistently)
  - This truth assignment is consistent and all clauses are satisfied

SATISFIABILITY

- Claim: 3-SAT $\leq_P$ INDEPENDENT SET
- Proof: Let $x_1, \ldots, x_n$ be a satisfying assignment
  - Select one true literal from each clause/triangle
  - This is an independent set of size k

TRANSITIVITY

- If $X \leq_P Y$ and $Y \leq_P Z$ then $X \leq_P Z$
- Thus, we have shown that 3-SAT$\leq_P$ INDEPENDENT SET $\leq_P$VERTEX COVER $\leq_P$ SET COVER

SUMMARY

- Defined Cook reductions
- Saw some simple reduction strategies
- 3-SAT$\leq_P$ INDEPENDENT SET $\leq_P$VERTEX COVER $\leq_P$ SET COVER
- 8.1 and 8.2 in KT