COL 351: Analysis and Design of Algorithms

Lecture 35-36

Vertex Cover

Given: A graph $G = (V, E)$ with n vertices.

Def: A subset $W \subseteq V$ such that for each edge $(a, b) \in E$, either a or b lies in W.

Decision Version:
Find if there is a vertex-cover of size $\leq k$.

Verifier($(G, k), S$)

1. If $|S| \gneq k$:
   Return *False*

2. For each $e = (u, v) \in E$:
   If both $u,\ v$ not in S:
   Return *False*

3. Return *True*

Dominating Set

Given: A graph $G = (V, E)$ with n vertices.

Def: A subset $D \subseteq V$ such that for each $v \notin D$, a neighbour of v lies in set "D".

Decision Version:
Find if there is a dominating-set of size $\leq k$.

Verifier($(G, k), S$)

1. If $|S| \gneq k$:
   Return *False*

2. For each $v \in (V \setminus S)$:
   If $N(v) \cap S = \varnothing$:
   Return *False*

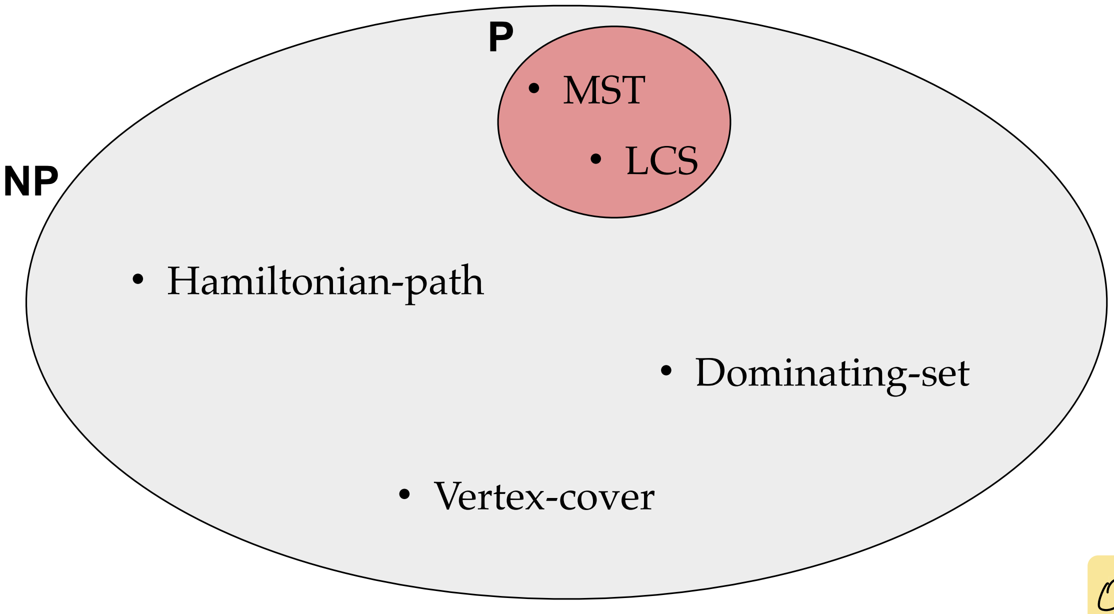
3. Return *True*

NP Class

- The class of ALL <u>decision</u> problems which have **Polynomial-time Verifier**.

P Class

- The class of ALL <u>decision</u> problems which have **Polynomial-time algorithm**.



Open Problem:
Is $P = NP$?

NP Class

A "Decision-problem" X is said to be in NP iff for every instance I of problem X:

- There is a polynomial time algorithm/verifier $\boldsymbol{A}$ with output {yes,no} satisfying

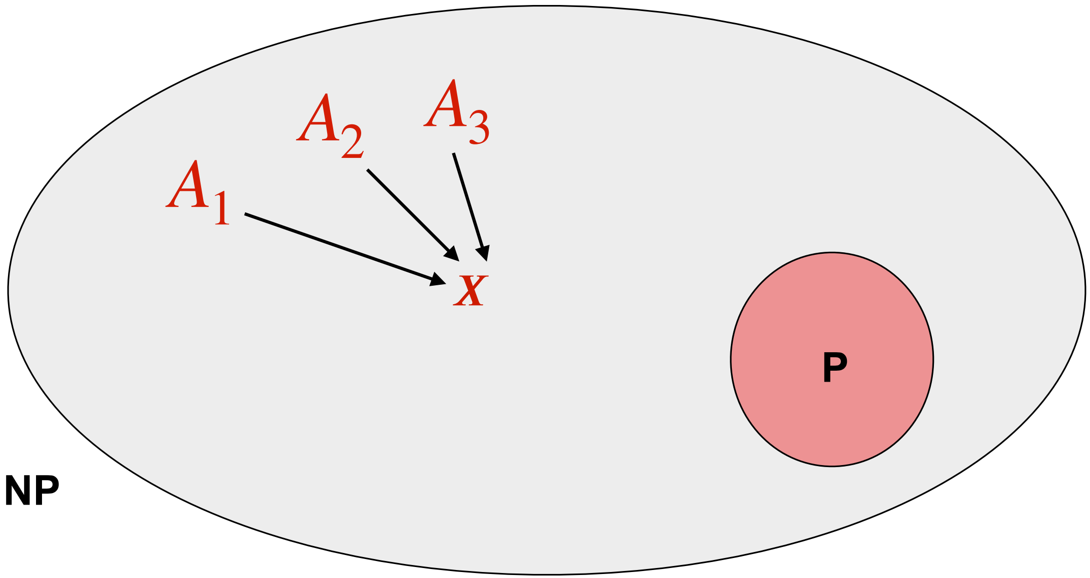
  For any proposed **certificate** "S" of length $O(|I|^c)$,
  $\boldsymbol{A}$ runs in $O(|I|^d)$ time over (I, S) to check if S is a valid solution to I.

- If I is "YES"-instance, then there exists an "S" of length $O(|I|^c)$ such that $\boldsymbol{A}(I,S)$ = YES

- If I is "NO"-instance, then for all "S" of length $O(|I|^c)$ $\boldsymbol{A}(I, S)$ = NO

NP-Complete problem — Hardest problem in NP class

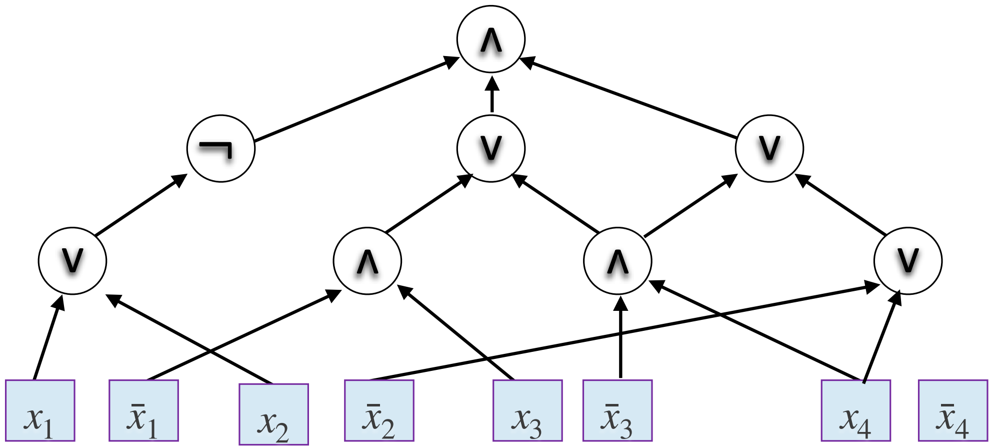
Definition: A problem X in NP class is **NP-Complete** if for every $A \in$ NP, we have $A \leq_P X$



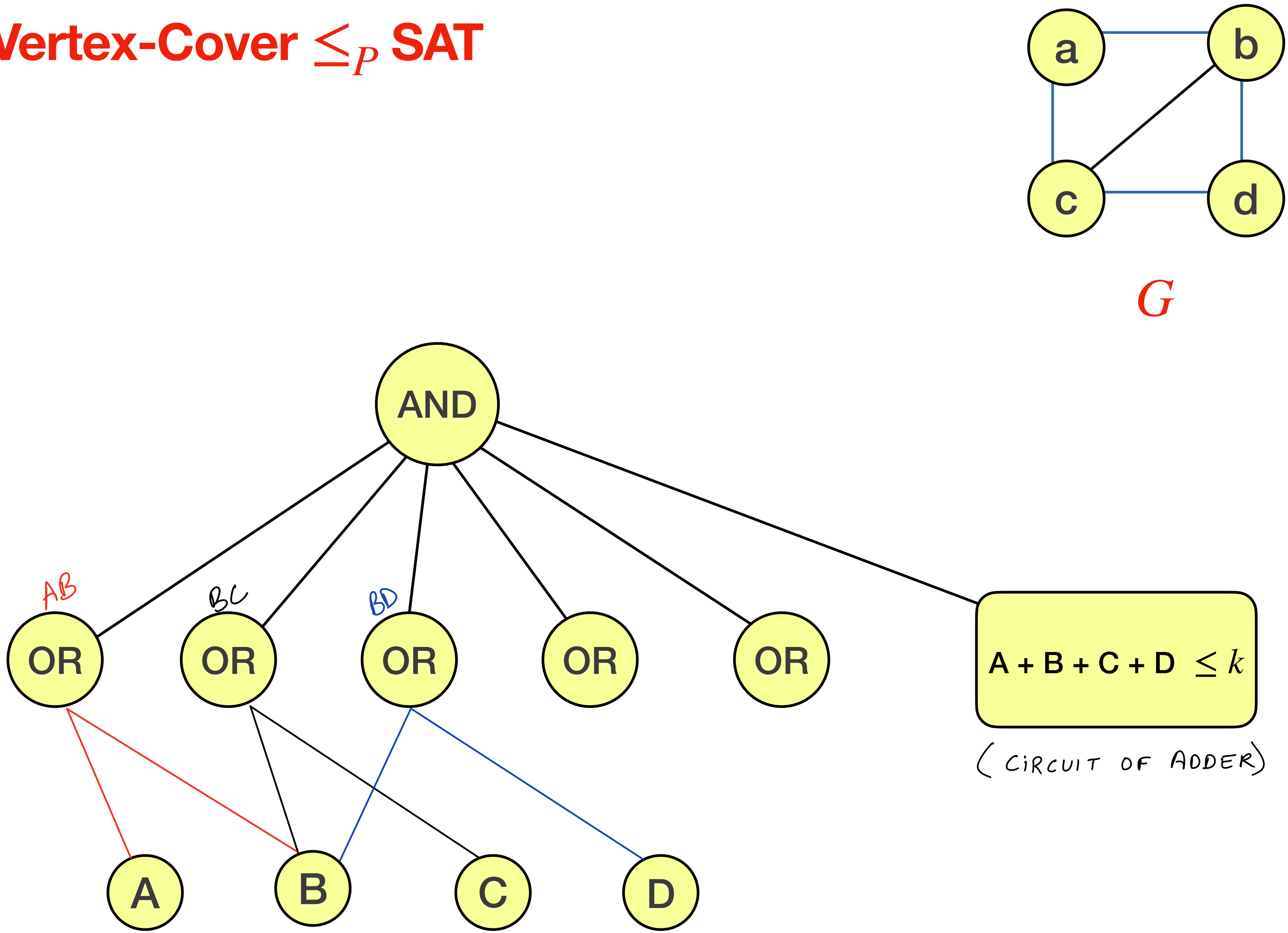
SAT (Circuit-Satisfiability problem)

Given: A DAG with nodes corresponding to AND, NOT, OR gates and n boolean inputs.

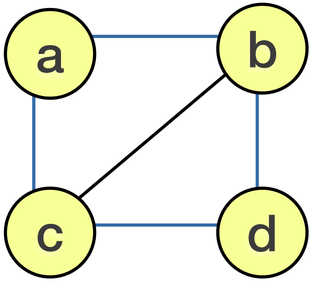
Problem: Is there an assignment of n inputs which gives output 1.



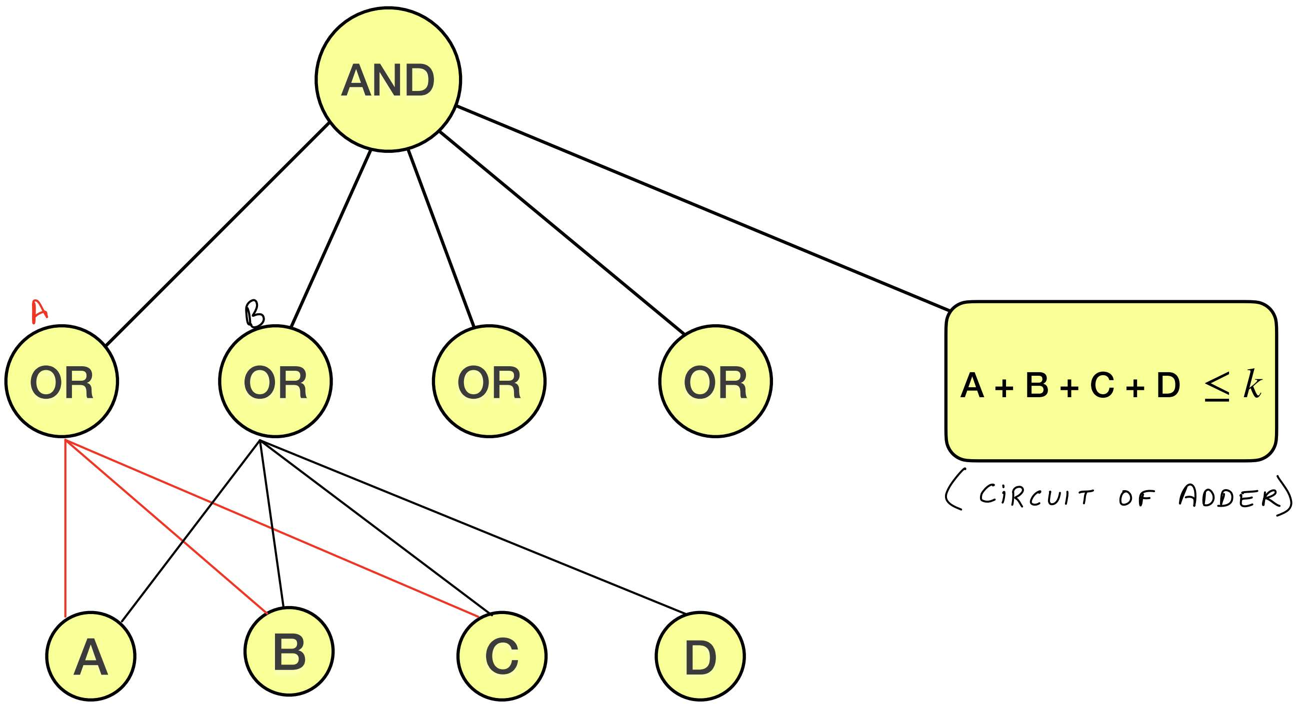
Vertex-Cover $\leq_P$ SAT



Dominating-Set $\leq_P$ SAT

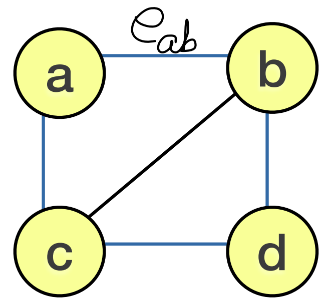


G

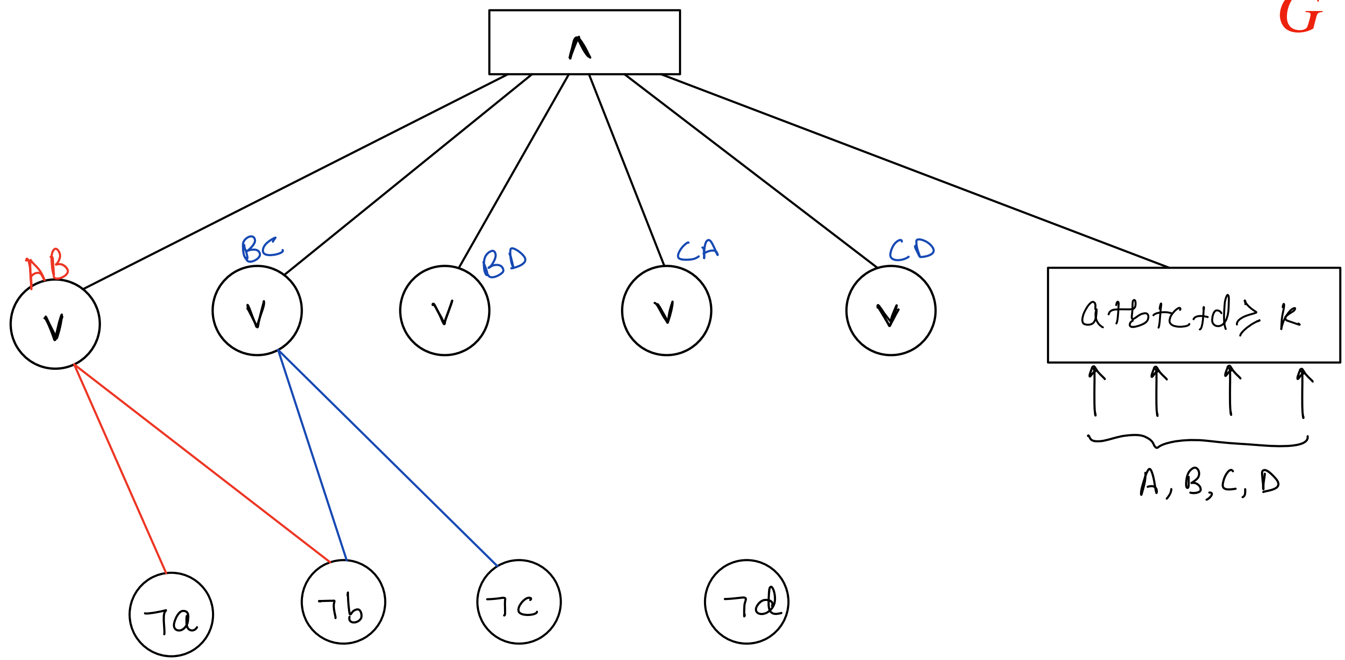


Independent-Set $\leq_P$ SAT

A set S is called Independent if there is no edge in G with both endpoints in S.



G



Cook Levin Theorem — SAT is NP-complete

Note: This is high level intuition, actual proof is quite involved & not part of syllabus

Theorem 1: For any problem X in NP-class, $X \leq_P$ SAT

Proof Sketch:

VertexCover-Verifier $((G, k), S)$

1. If $|S| \gneq k$:
   Return *False*

2. For each $e = (u, v) \in E$:
   If both u, v not in S:
   Return *False*

3. Return *True*

Algorithm A

Poly(n)

Verifier $((G, k), S)$

Re write algo steps without loops, jumps, Fn calls, etc.

New algorithm A'
(without for loops/jumps)

size = poly in input

Poly(n)

SAT instance

3-SAT (3-Circuit-Satisfiability problem)

Given: A SAT which is an AND of clauses containing 3 literals.

(A clause is just OR of literals).

Problem: Is there an assignment of ***n*** inputs which gives output 1.

Example:

$$C = (x_1 \vee \bar{x}_2 \vee x_3) \ \wedge \ (\bar{x}_4 \vee \bar{x}_2 \vee x_1) \ \wedge \ (x_2 \vee \bar{x}_3 \vee x_1) \ \wedge \ (x_4 \vee \bar{x}_3 \vee x_1)$$

Theorem 1: For each problem X in NP we have: $X \ \leq_P$ SAT.

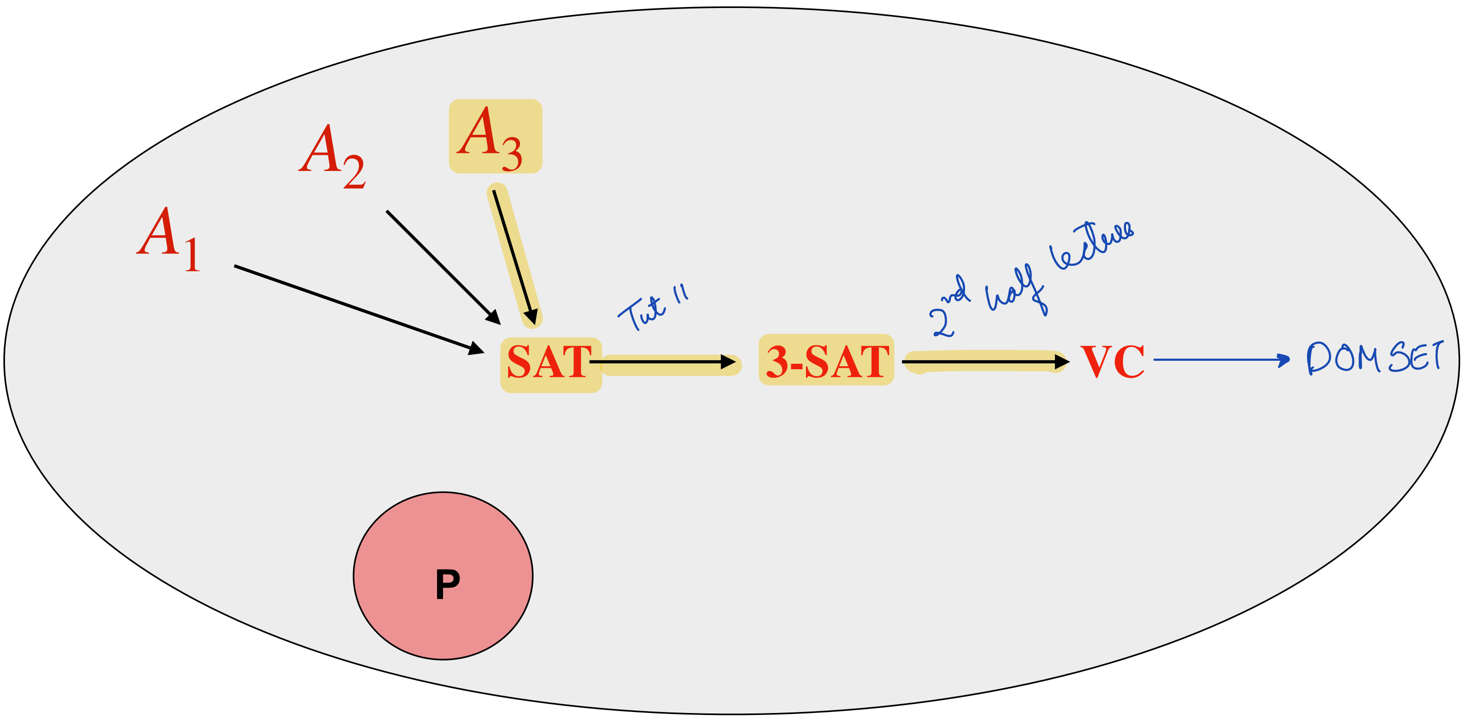
Theorem 2: SAT $\leq_P$ 3-SAT.

Theorem 2: SAT $\leq_P$ 3SAT

Proof: Homework (Tutorial 11)

- Step 1: Push negations to literals by De-Morgan's Law:
- Step 2: Make transformation to get a CNF.
- Step 3: Transform CNF into 3-SAT by introducing new variables.

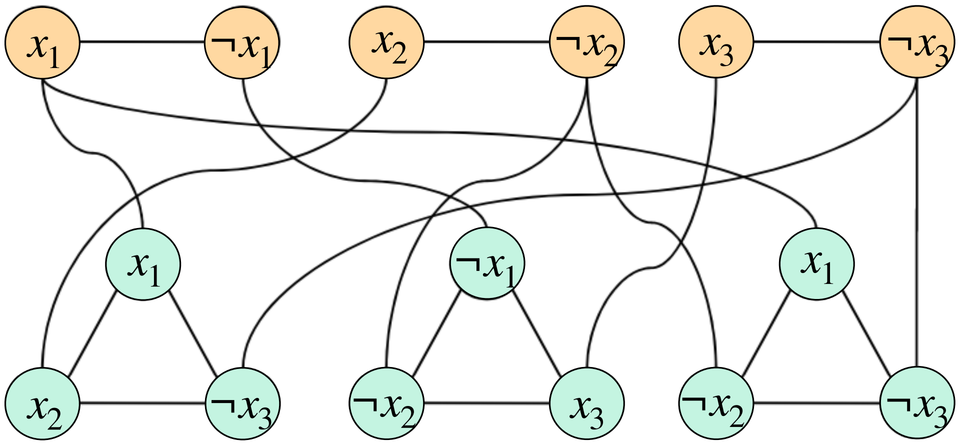
NP-Complete problems



NP class

3-SAT $\leq_P$ VC

- Graph corresponding to **3-SAT** instance: $(x_1 \vee x_2 \vee \neg x_3) \wedge (\neg x_1 \vee \neg x_2 \vee x_3) \wedge (x_1 \vee \neg x_2 \vee \neg x_3)$



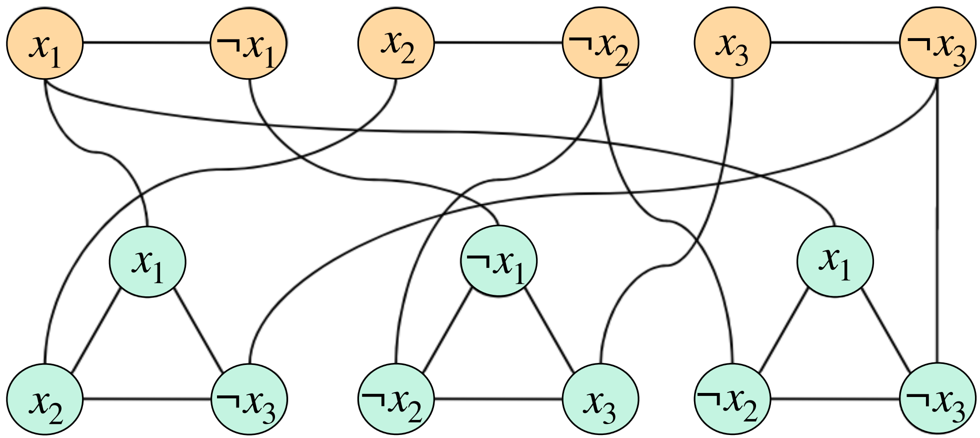
3-SAT instance ϕ

n variables, m clauses

Vertex-cover instance G_ϕ

$2n + 3m$ vertices, and $k = n + 2m$

Claim 1: If 3-SAT instance ϕ is satisfiable, then G_ϕ has a vertex cover of size $k = n + 2m$

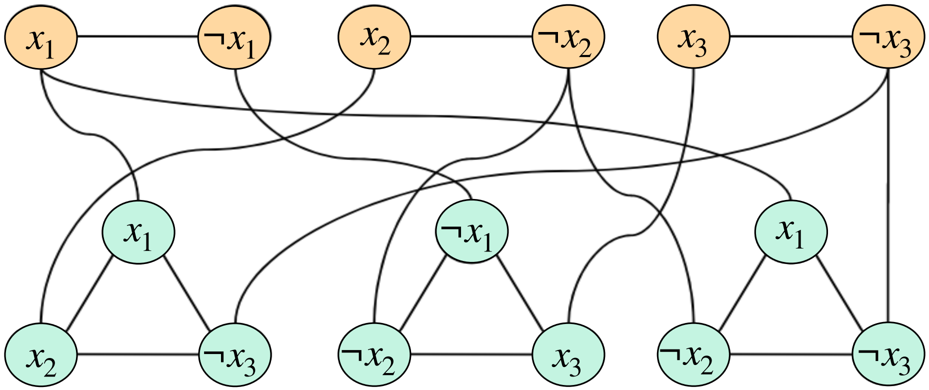


Proof:

Top Layer: For $i \in [1, n]$ if x_i =1, then choose x_i in vertex-cover, else choose $\neg x_i$ in vertex-cover.

Bottom Triangles: In each clause, ∃ at most two literals with “0” value, we put them in vertex-cover.

Claim 2: If G_ϕ has a vertex cover of size $k = n + 2m$, then 3-SAT instance ϕ is satisfiable.



Proof: Set $x_i = 1$ iff x_i is in VC in top-layer.

A vertex cover of size $k = n + 2m$ will satisfy:

- Top Layer: For $i \in [1, n]$ exactly one of x_i and $\neg x_i$ is chosen in VC.
- Bottom Triangles: In each clause, exactly two literals must be chosen in VC. In a triangle, if a literal.

  x_j (or $\neg x_j$) is not in VC, then corresponding variable x_j (or $\neg x_j$**)** must be 1. This ensures satisfiability.

Some NP Complete Problems