CSE 331: Algorithms & Complexity

“K-Colorings”

Prof. Charlie Anne Carlson (She/Her)

Lecture 38

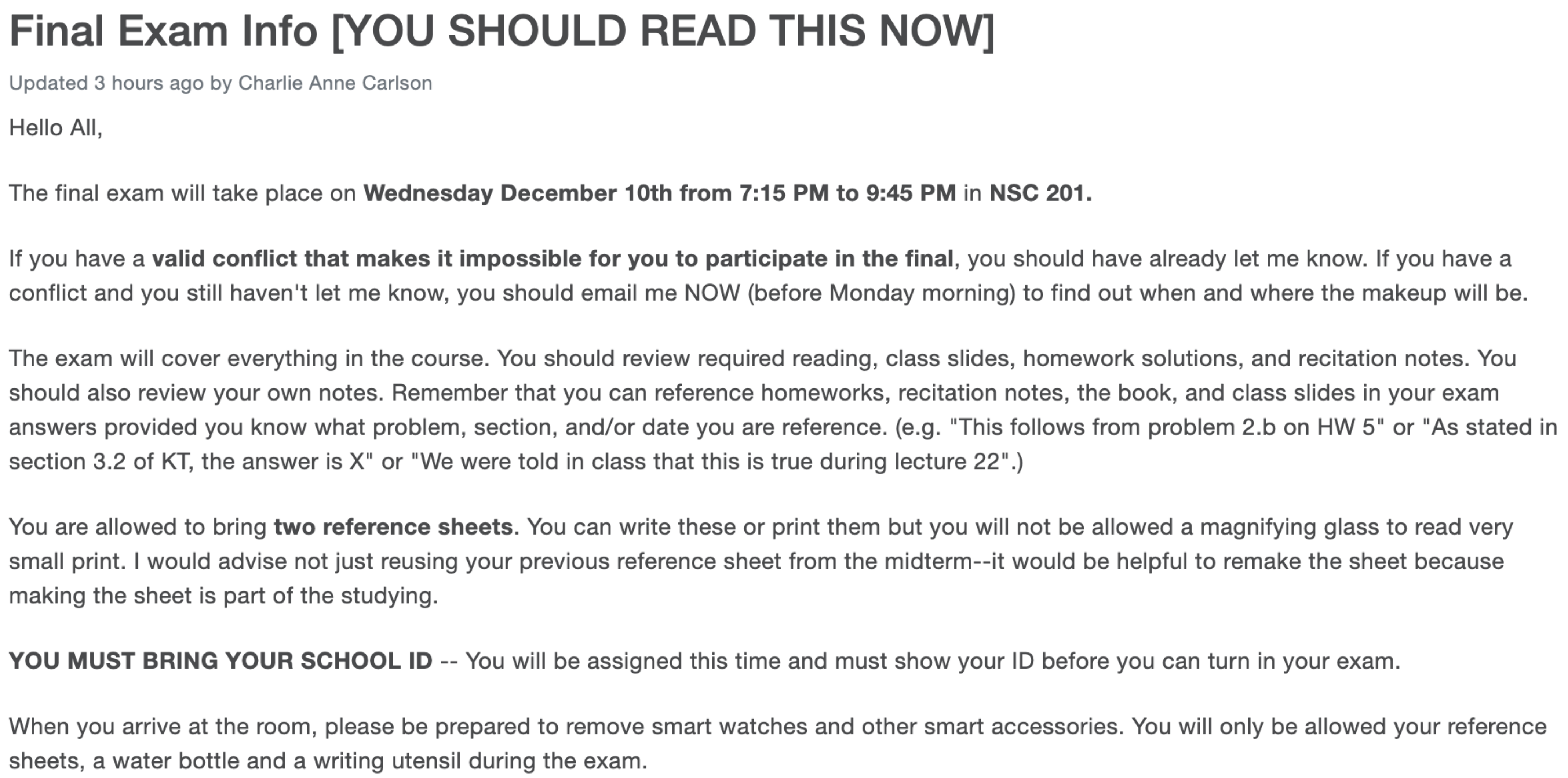
Dec 5th 2025

University at Buffalo

Course Evals

- Course evals must be submitted before December 10th.
  - They are due on December 9th at 11:59 PM
- Please consider filling them out.
  - Be honest!

Final Exam Post



Final Exam Info [YOU SHOULD READ THIS NOW]

Updated 3 hours ago by Charlie Anne Carlson

Hello All,

The final exam will take place on **Wednesday December 10th from 7:15 PM to 9:45 PM** in **NSC 201.**

If you have a **valid conflict that makes it impossible for you to participate in the final**, you should have already let me know. If you have a conflict and you still haven't let me know, you should email me NOW (before Monday morning) to find out when and where the makeup will be.

The exam will cover everything in the course. You should review required reading, class slides, homework solutions, and recitation notes. You should also review your own notes. Remember that you can reference homeworks, recitation notes, the book, and class slides in your exam answers provided you know what problem, section, and/or date you are reference. (e.g. "This follows from problem 2.b on HW 5" or "As stated in section 3.2 of KT, the answer is X" or "We were told in class that this is true during lecture 22".)

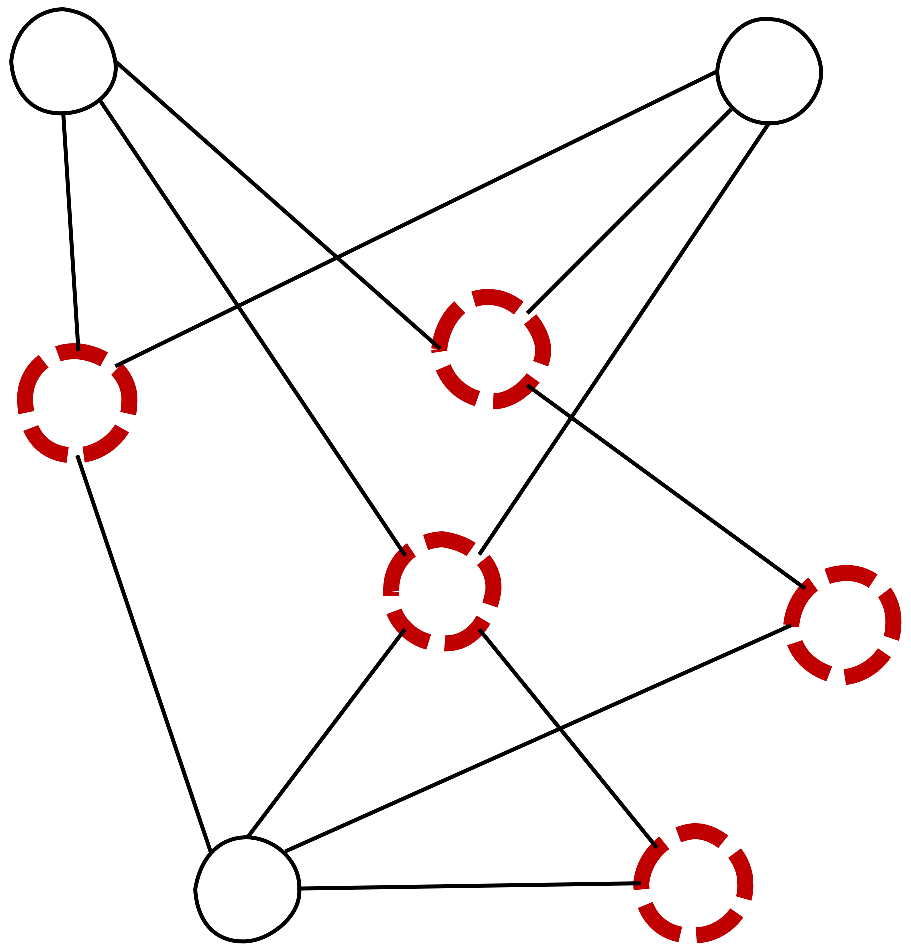
You are allowed to bring **two reference sheets**. You can write these or print them but you will not be allowed a magnifying glass to read very small print. I would advise not just reusing your previous reference sheet from the midterm--it would be helpful to remake the sheet because making the sheet is part of the studying.

YOU MUST BRING YOUR SCHOOL ID -- You will be assigned this time and must show your ID before you can turn in your exam.

When you arrive at the room, please be prepared to remove smart watches and other smart accessories. You will only be allowed your reference sheets, a water bottle and a writing utensil during the exam.

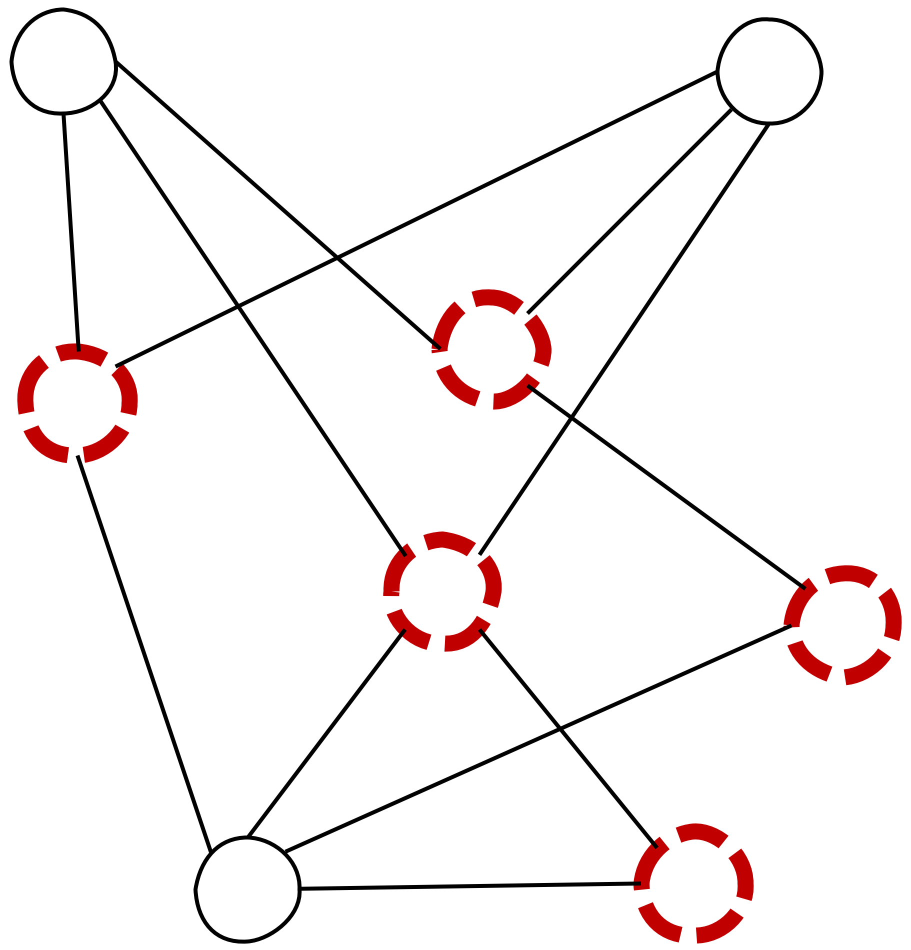
NP-complete

- To show that a problem is NP-complete, you must do the following:
  - Show the problem is in NP
  - Show that the problem is has hard as any other problem in NP



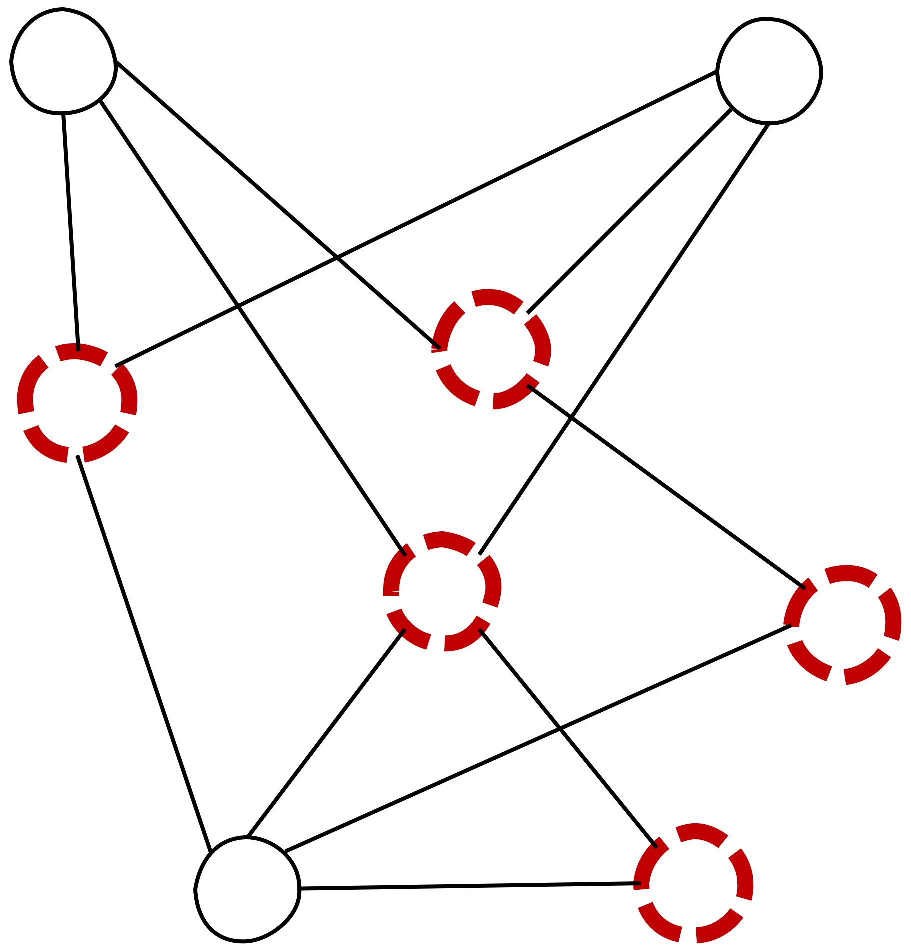
NP-complete

- To show that a problem is NP-complete, you must do the following:
  - Show that there exists a polynomial time verifier.
  - Show that you can reduce to it from a known NP-hard problem like 3-SAT.



NP-complete

- (Circuit) SAT
- 3-SAT
- Independent Set
- Clique
- Vertex Cover
- Set Cover
- K-coloring



3-SAT $\leq_p$ Independent Set

3-SAT

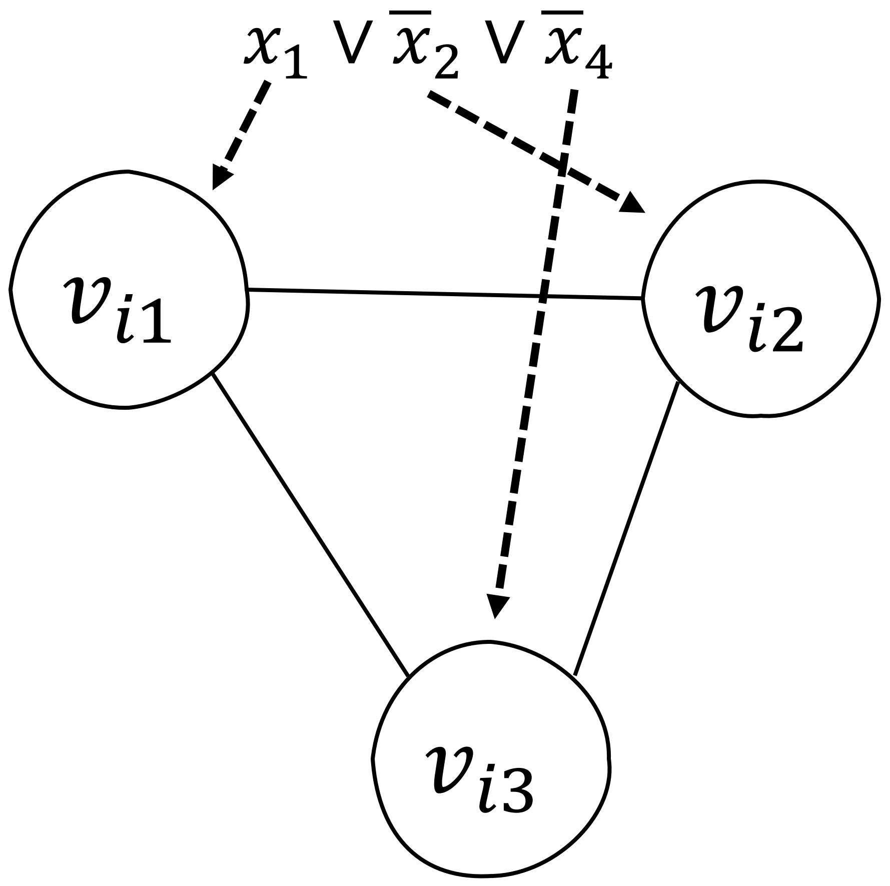
- **Input**: List of m clauses $C = C_1, \ldots, C_m$ over a set of n variables $X = x_1, \ldots, x_n$.
- **Output**: Does there exist an assignment to X such that each clause is satisfied.

Independent Set

- **Input**: A graph $\mathrm{G} = (\mathrm{V}, \mathrm{E})$ and integer k.
- **Output**: Does there exist an independent set in G of size at least k.

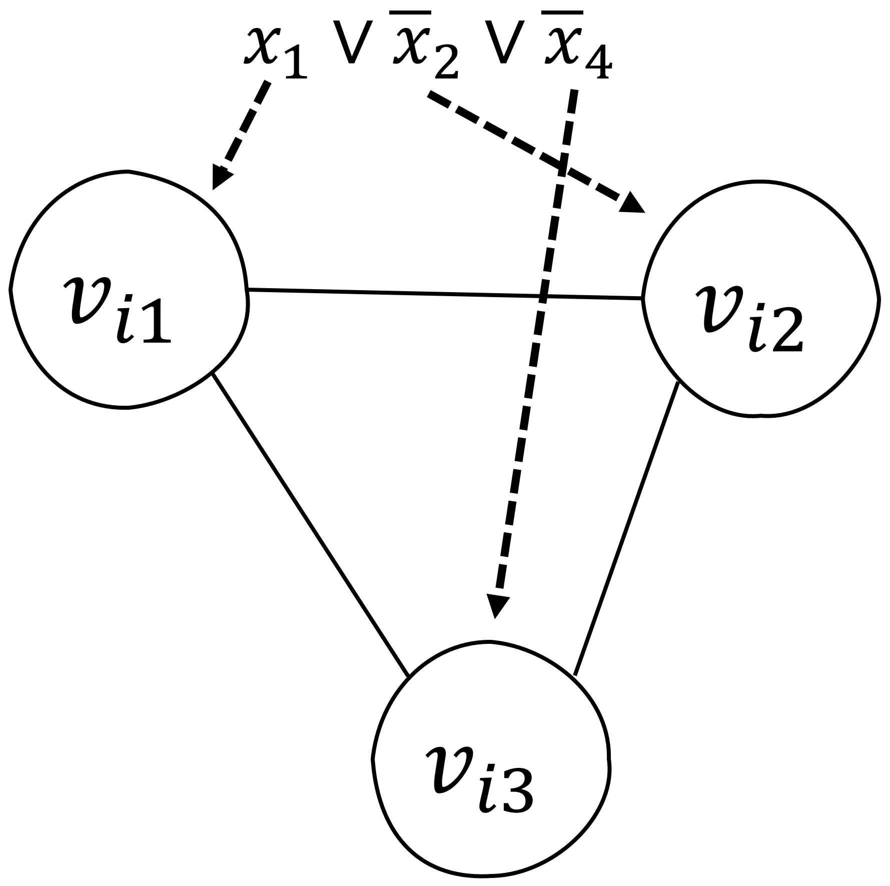
3-SAT $\leq_p$ Independent Set

- We need to find a way to convert our instance of 3-SAT into an instance of independent set.
- For each clause $C_i \in C$ we construct a triangle in G with vertices v_{i1}, v_{i2}, and v_{i3}.



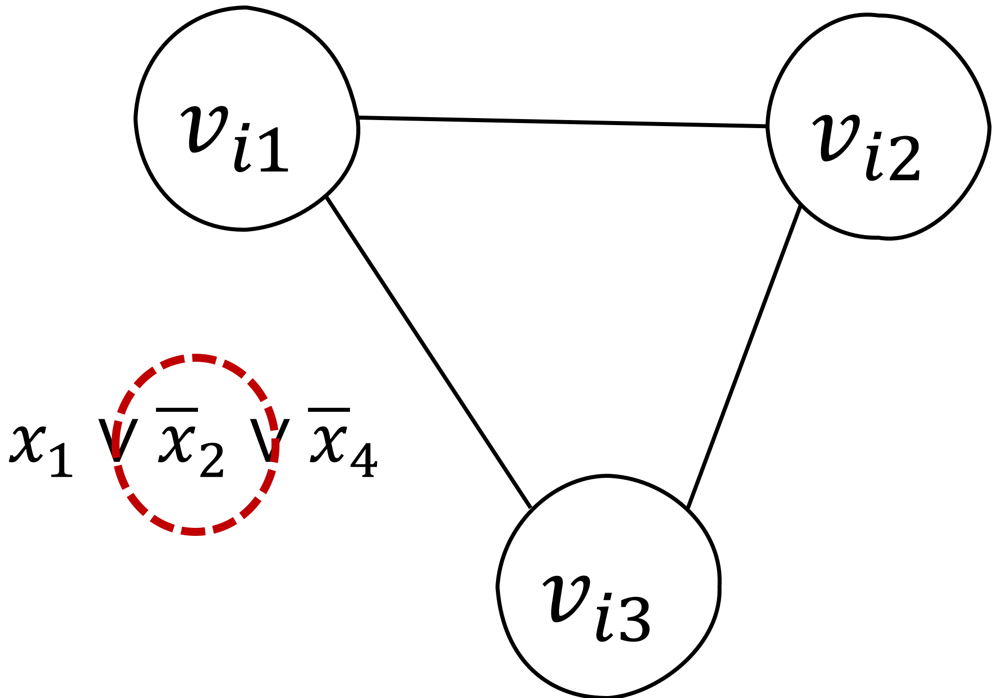
3-SAT $\leq_p$ Independent Set

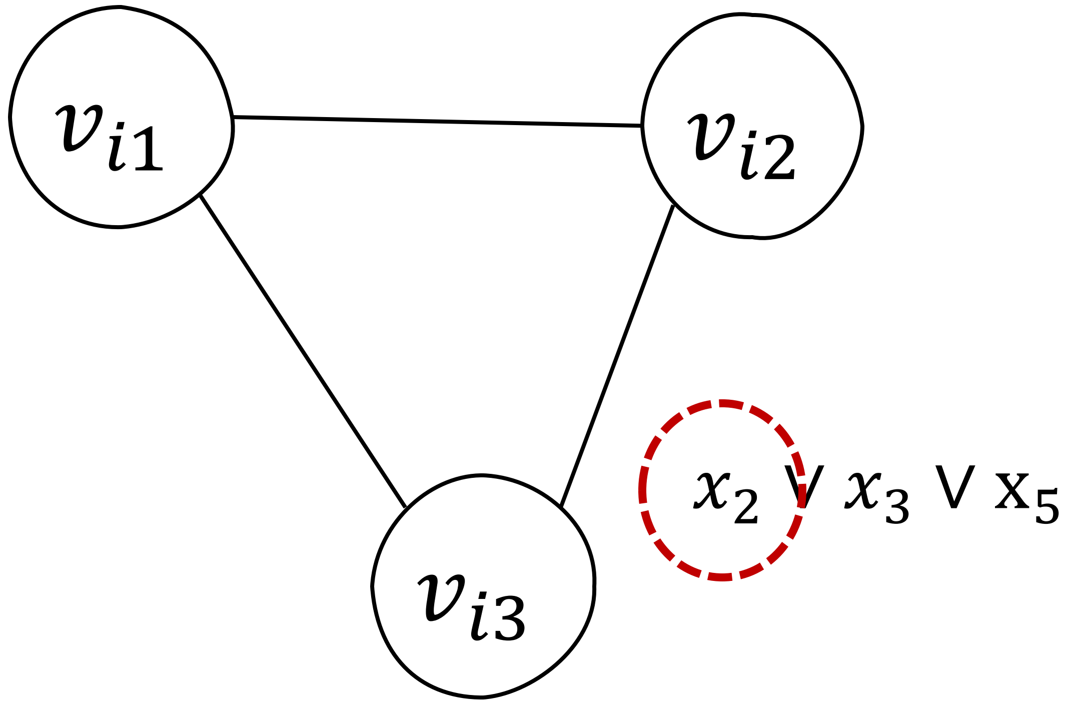
- Observe that if there is an independent set of size m in G, then it must pick a single vertex from each triangle.
- Think of this as saying that term it represents evaluates to true.



3-SAT $\leq_p$ Independent Set

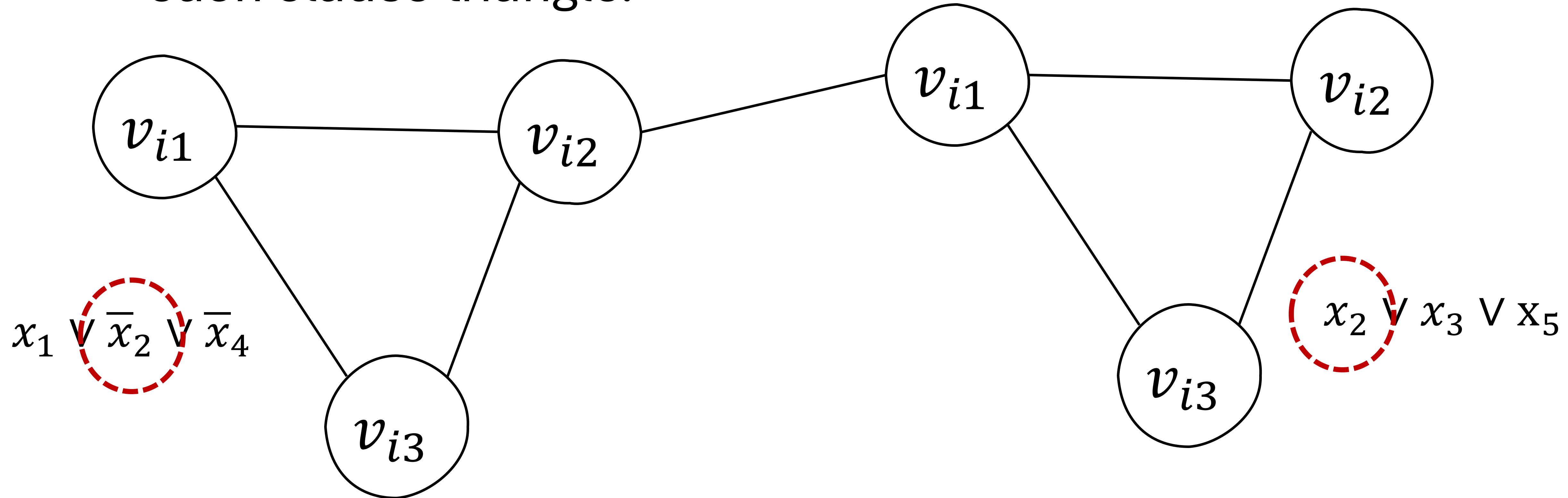
- If two clauses $C_i, C_j \in \mathrm{C}$ “disagree” on a variable x_q then add an edge between the two nodes that represent x_q in each clause triangle.





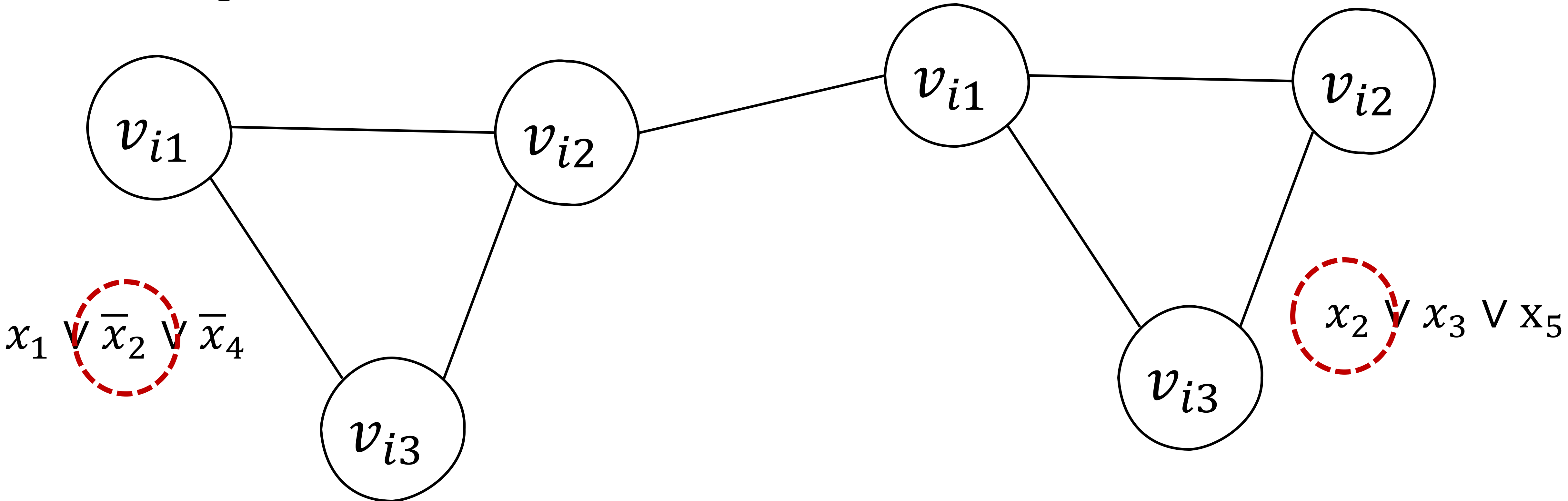
3-SAT $\leq_p$ Independent Set

- If two clauses $C_i, C_j \in \mathrm{C}$ “disagree” on a variable x_q then add an edge between the two nodes that represent x_q in each clause triangle.



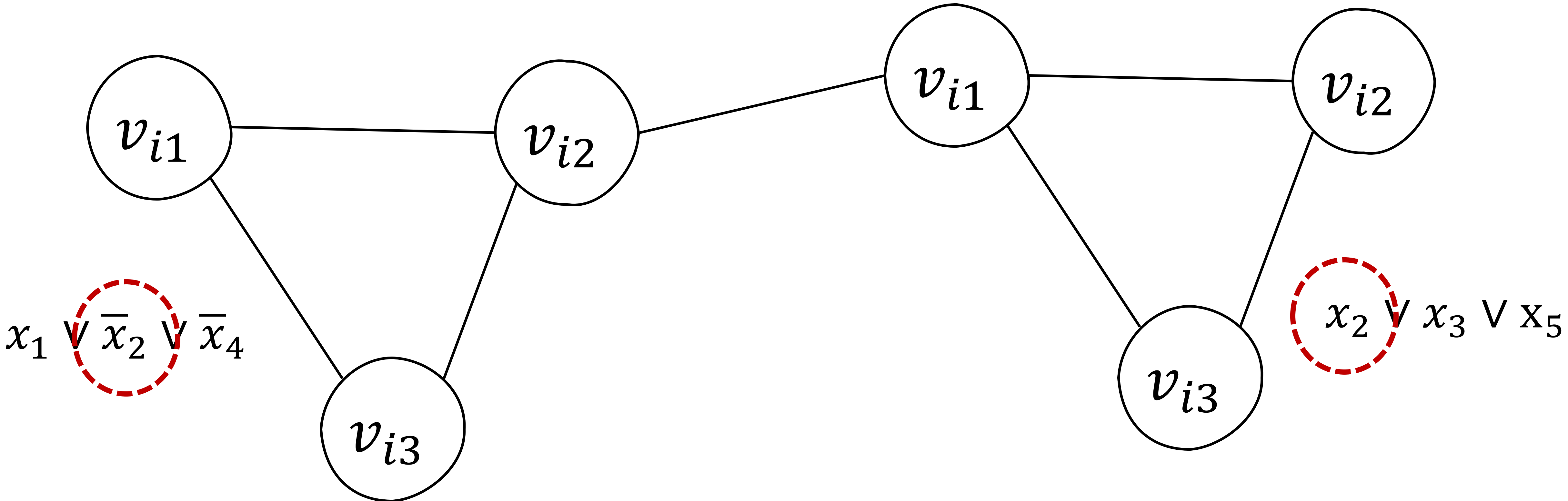
3-SAT $\leq_p$ Independent Set

- Observe that if there exists an independent set of size m (number of clause) then each triangle must contain a single vertex in the the set.



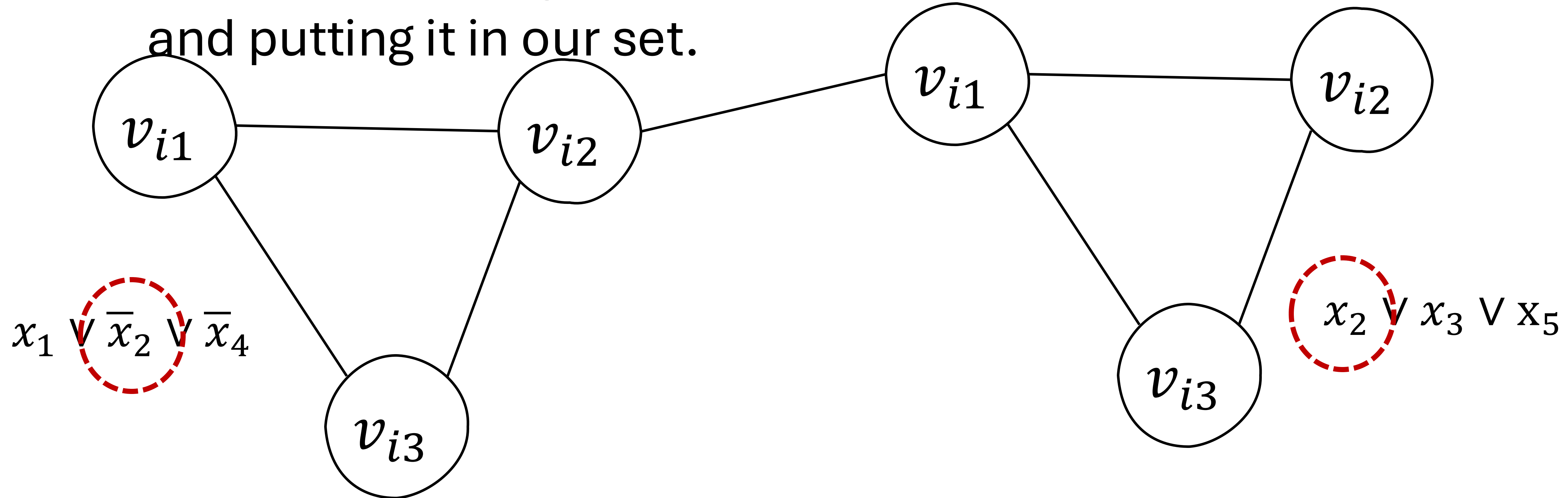
3-SAT $\leq_p$ Independent Set

- Moreover, for each variable $x_q \in X$ we the independent set can't contain a vertex that represents it being 0 and 1 at the same time.



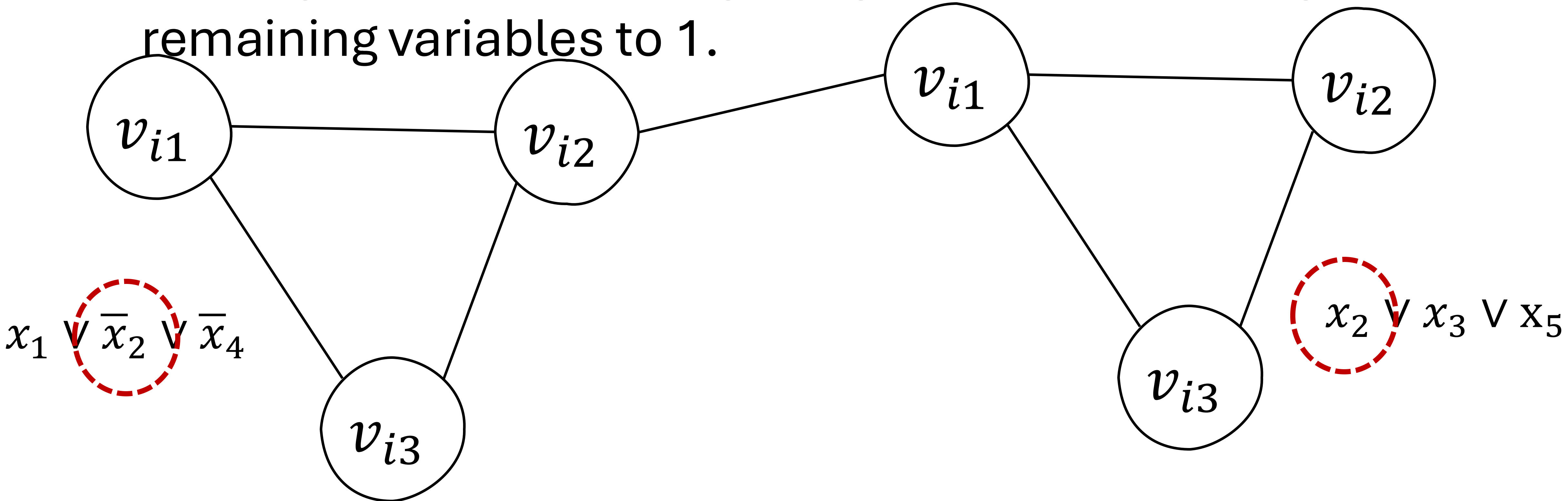
3-SAT $\leq_p$ Independent Set

- You can map a satisfying assignment to the 3-SAT formula to an independent set of size m in G by looking at each clause and taking the first term that evaluates to be true and putting it in our set.



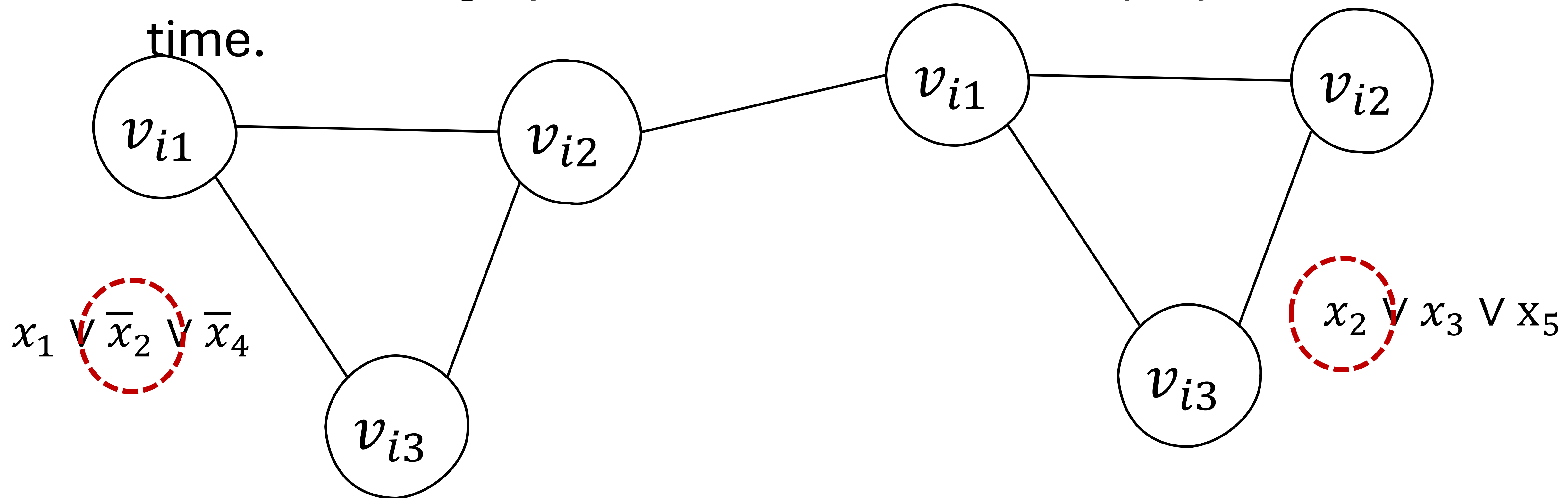
3-SAT $\leq_p$ Independent Set

- You can map an independent set in G to a satisfying assignment by looking each each vertex in the set, reading its corresponding assignment, and setting all remaining variables to 1.



3-SAT $\leq_p$ Independent Set

- Construct the graph as described above.
- Ask to find an independent set of size k = m.
- Observe this graph can be constructed in polynomial time.



Transitivity

Claim: If $Z \leq_p Y$, and $Y \leq_p X$, then $Z \leq_P X$.

Question: If I tell you that X is NP-hard. Then does there exist a polynomial-time reduction from X to Independent Set?

Question: If I tell you that X is NP-complete. Then does there exist a polynomial-time reduction reduction from Independent Set to X?

Transitivity

Claim: If $Z \leq_p Y$, and $Y \leq_p X$, then $Z \leq_P X$.

Question: If I tell you that X is NP-hard. Then does there exist a polynomial-time reduction from X to Independent Set?

Answer: No because NP-hard doesn't mean it is in NP.

Question: If I tell you that X is NP-complete. Then does there exist a polynomial-time reduction reduction from Independent Set to X?

Answer: Yes!

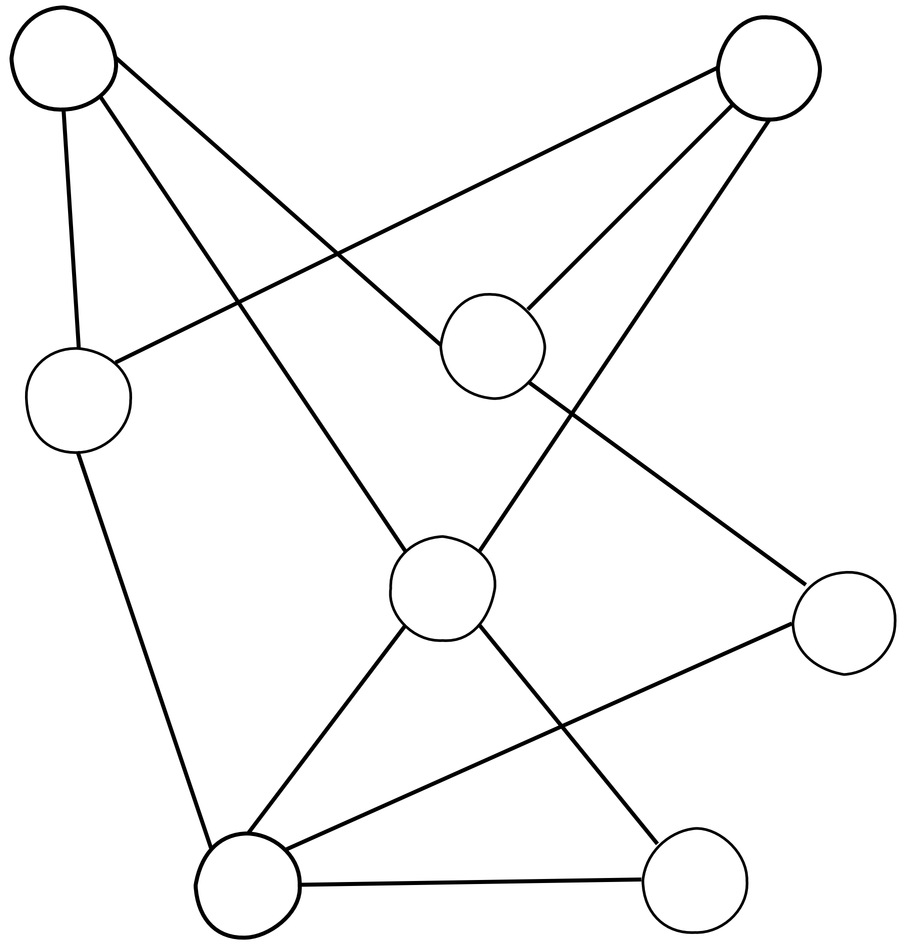
K-Coloring

Definition: Given an undirected graph G, a (proper) *k-coloring* of G is a function $\pi: V \rightarrow [k]$ such that for each edge $\{u, v\} \in E, \pi(u) \neq \pi(V)$.

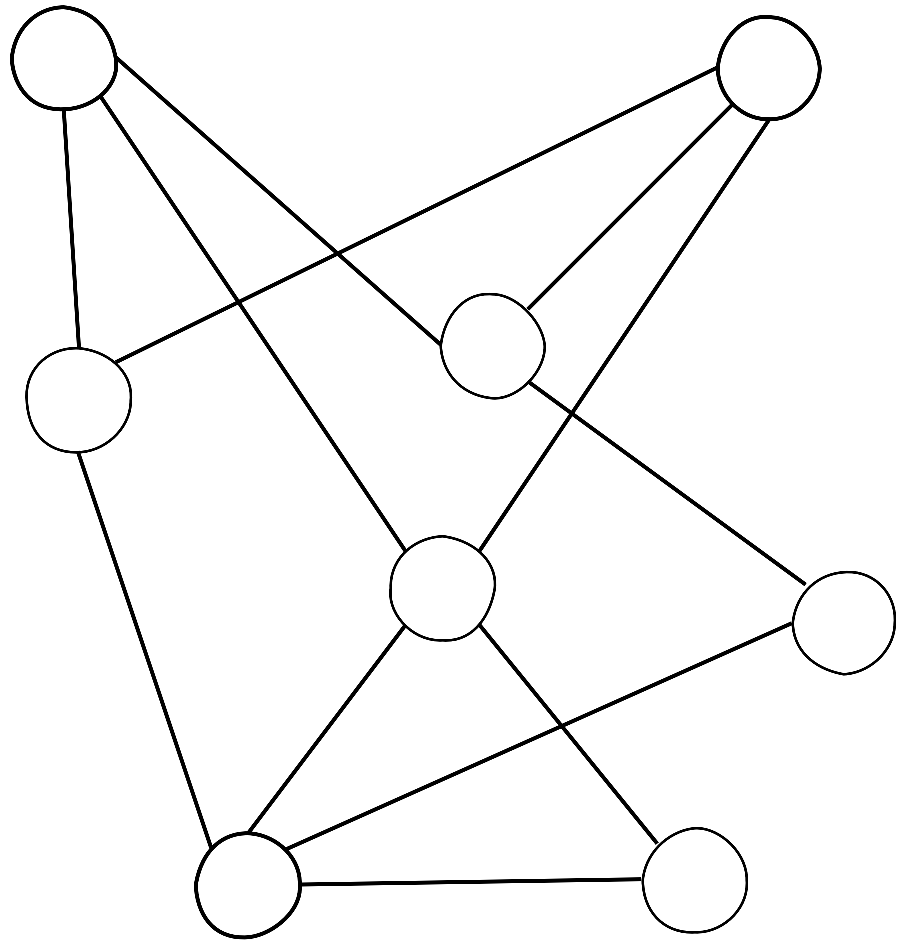
That is, it is an assignment to each vertex of a color from a set of k colors such that each edge is bichromatic.

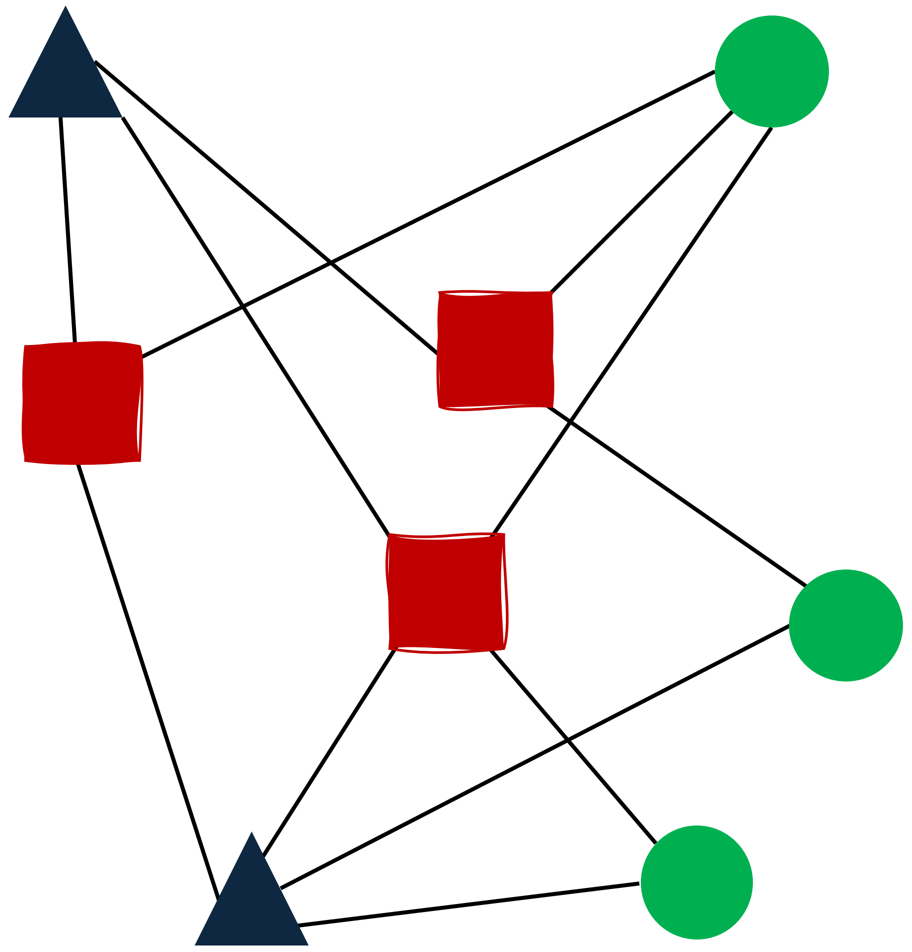
Definition: We say that a graph G is *k-colorable* if there exists a k-coloring of G.

3-Coloring

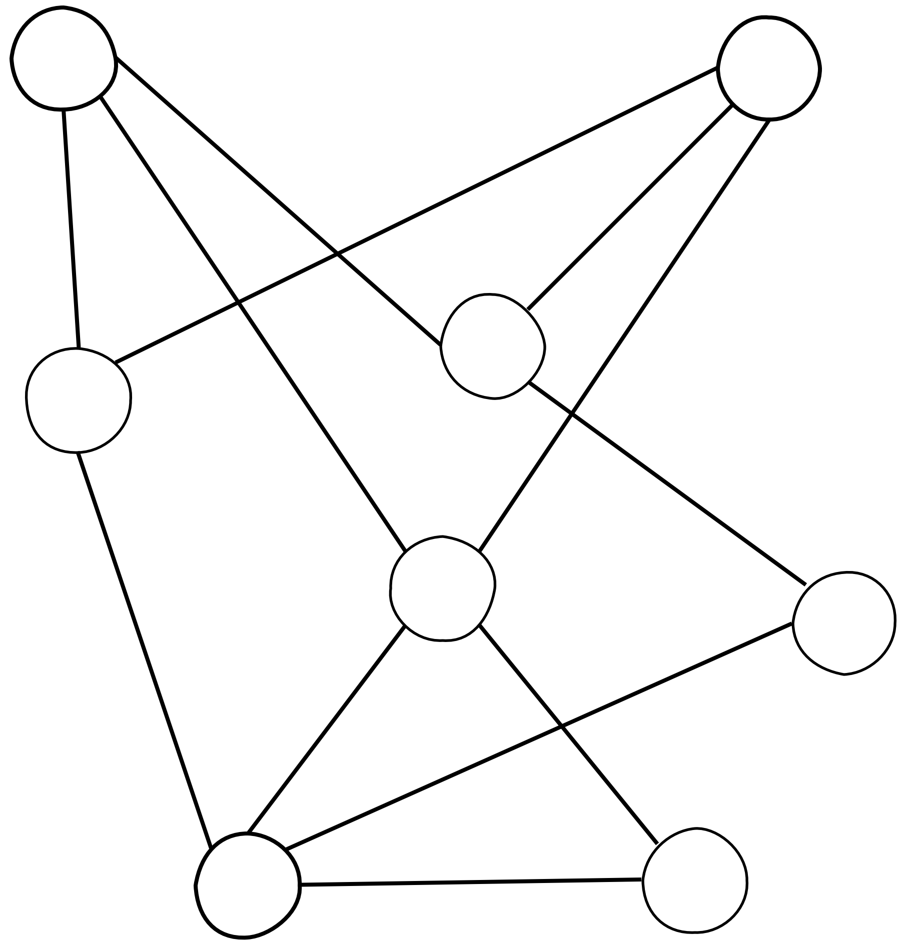


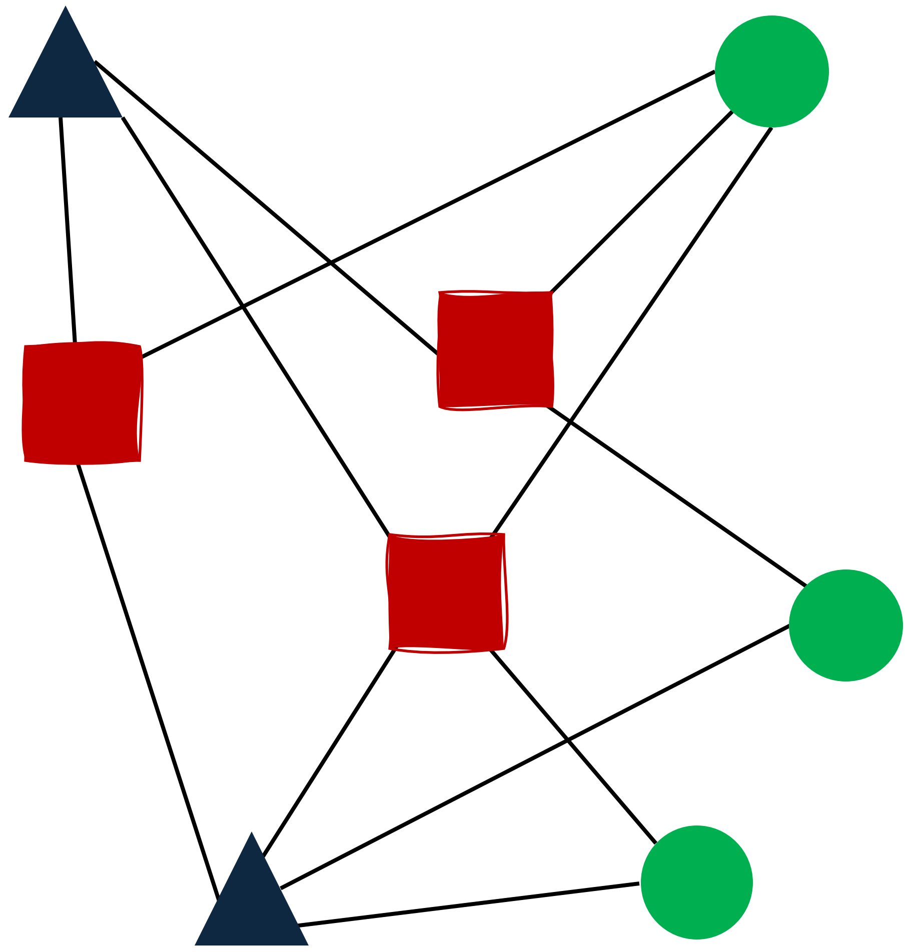
3-Coloring



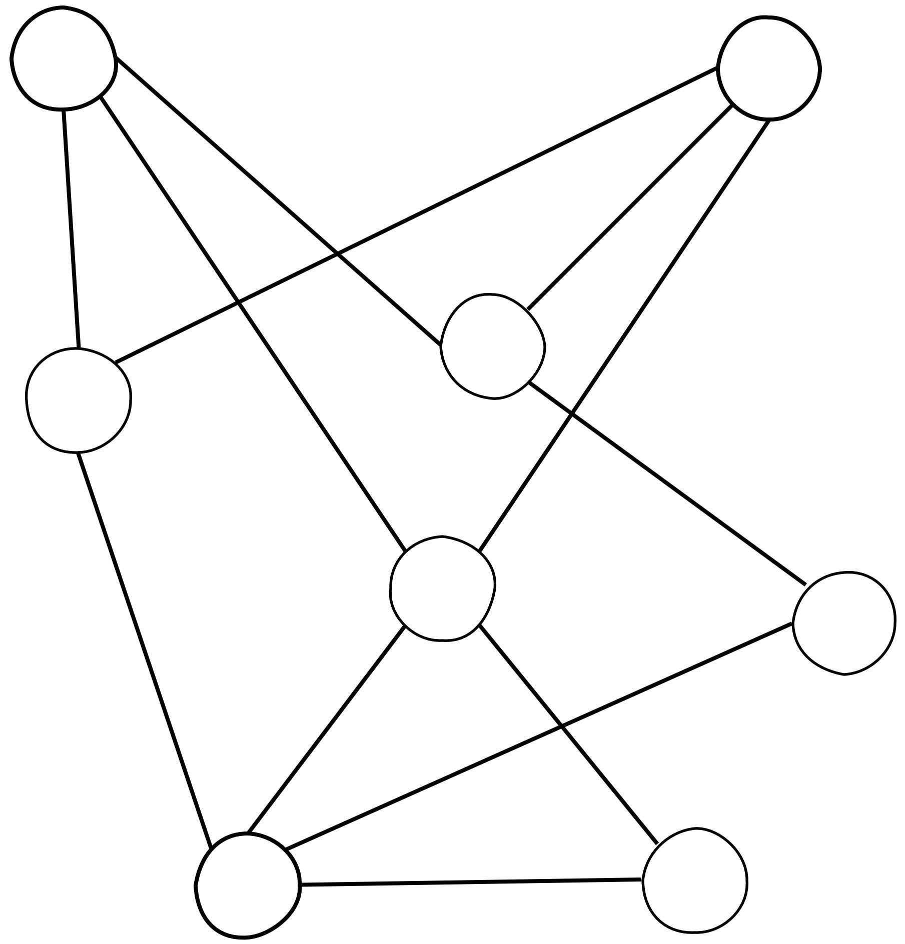


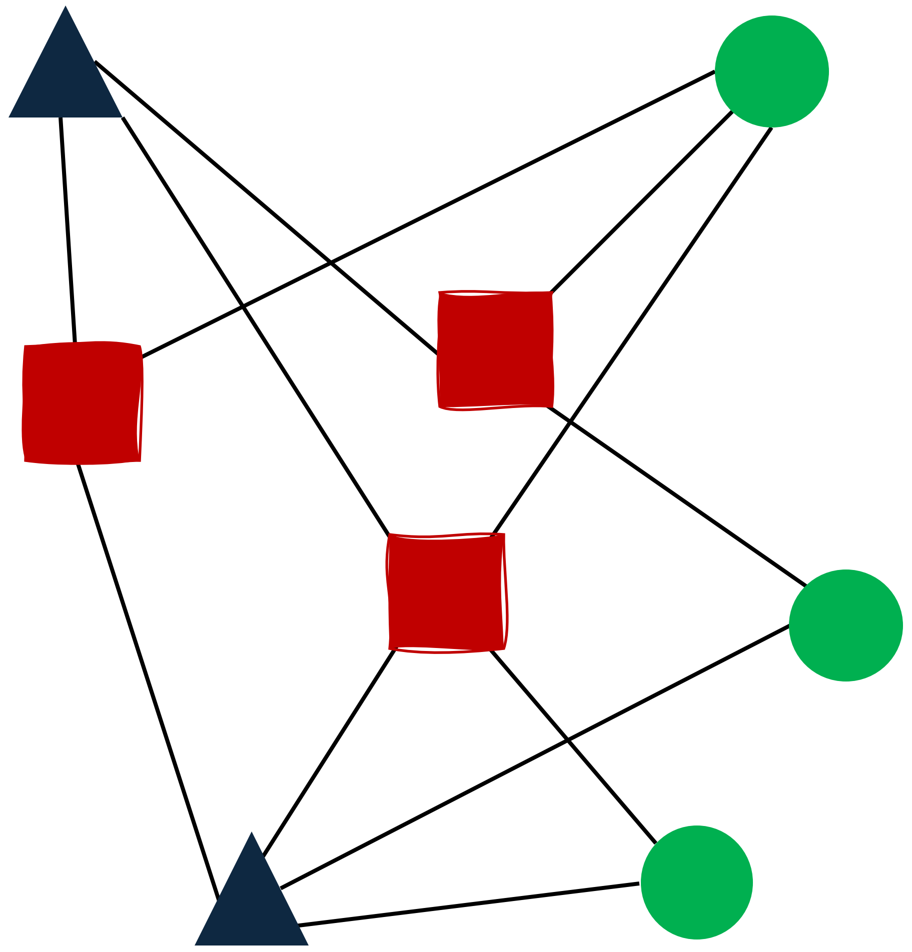
4-Coloring



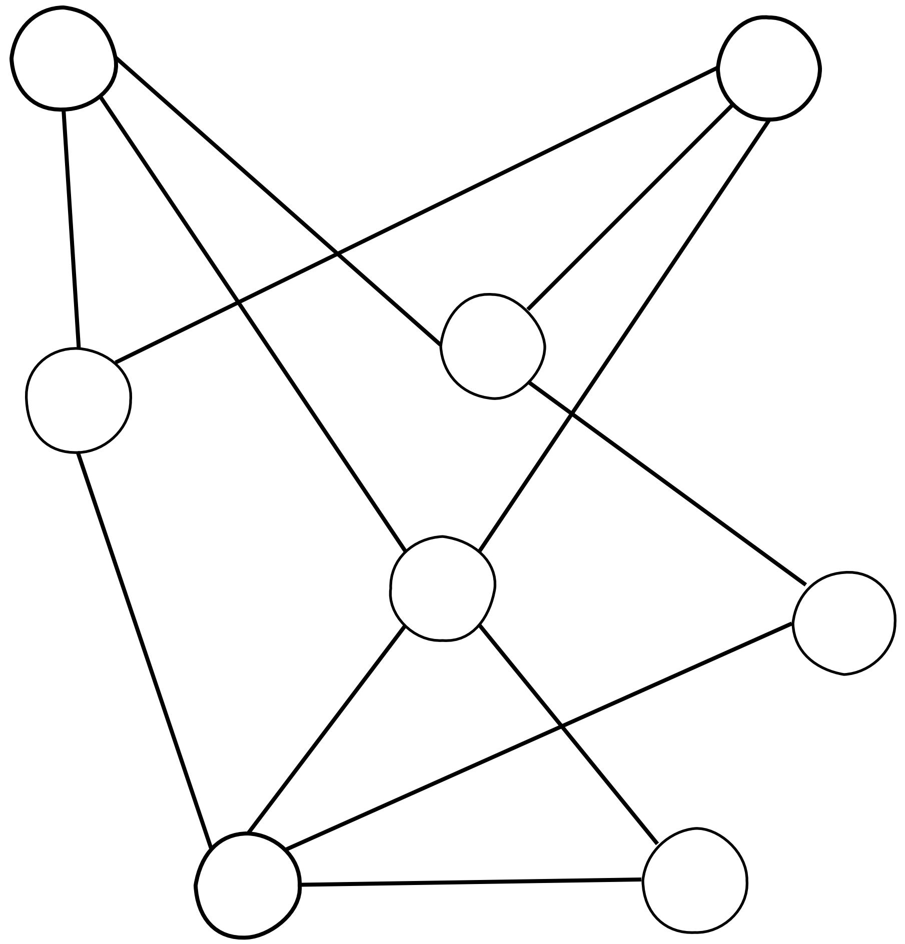


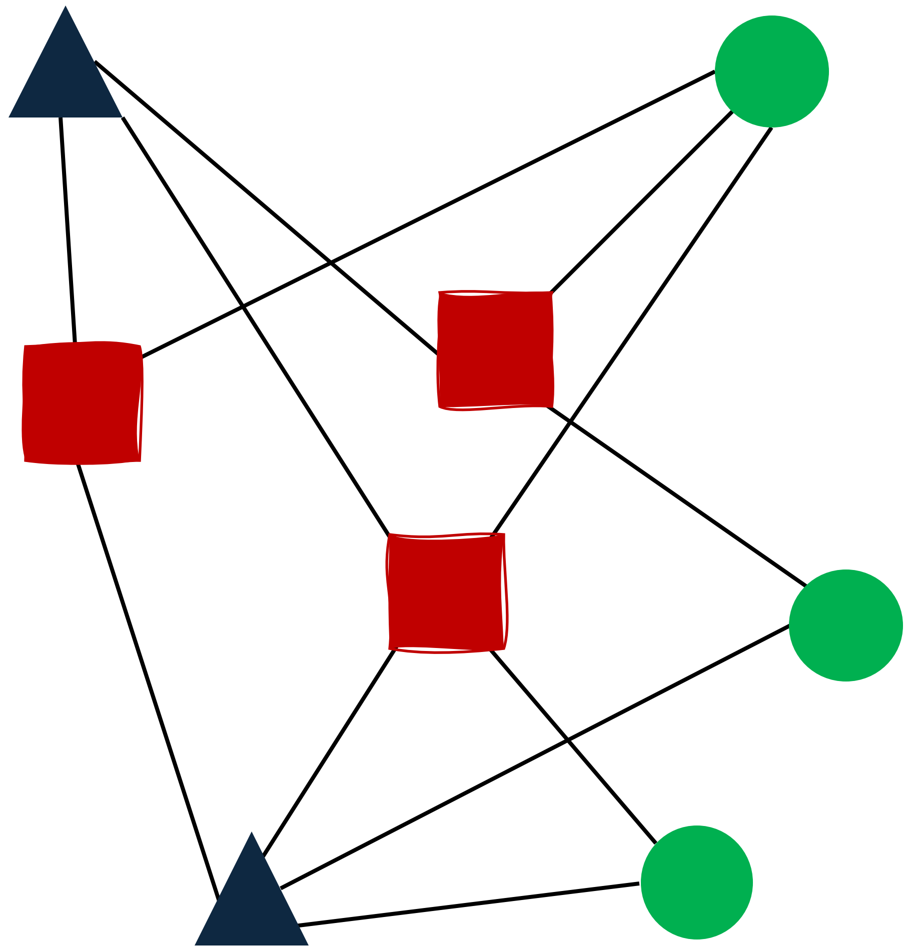
100-Coloring





No 2-Coloring

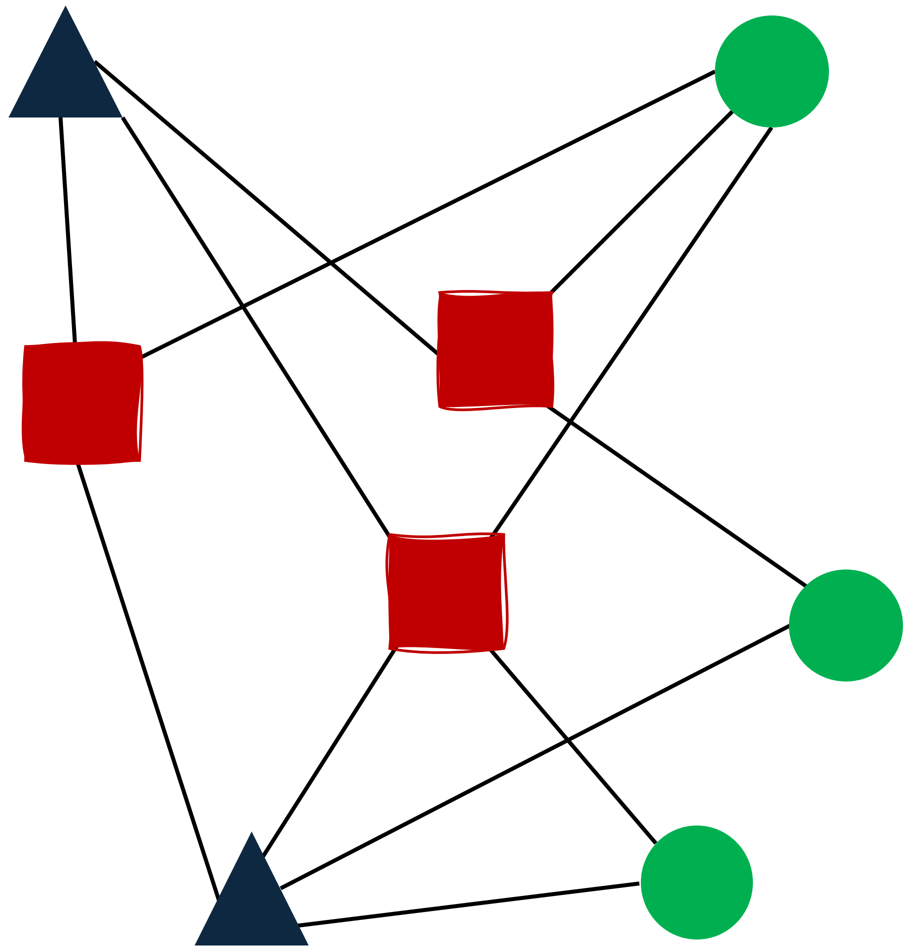




K-coloring

K-Coloring

- **Input**: Graph G and integer k.
- **Output**: Does there exist a k-coloring of G.

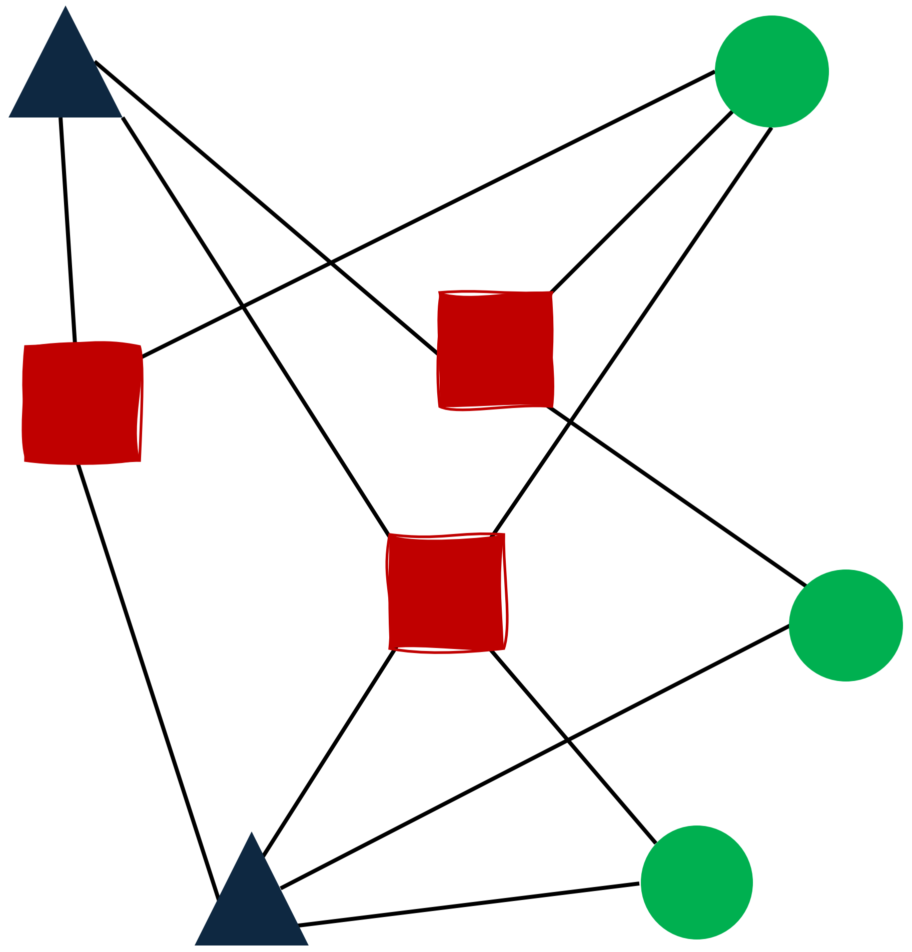


K-coloring

K-Coloring

- **Input**: Graph G and integer k.
- **Output**: Does there exist a k-coloring of G.

This in NP because you can check a coloring.

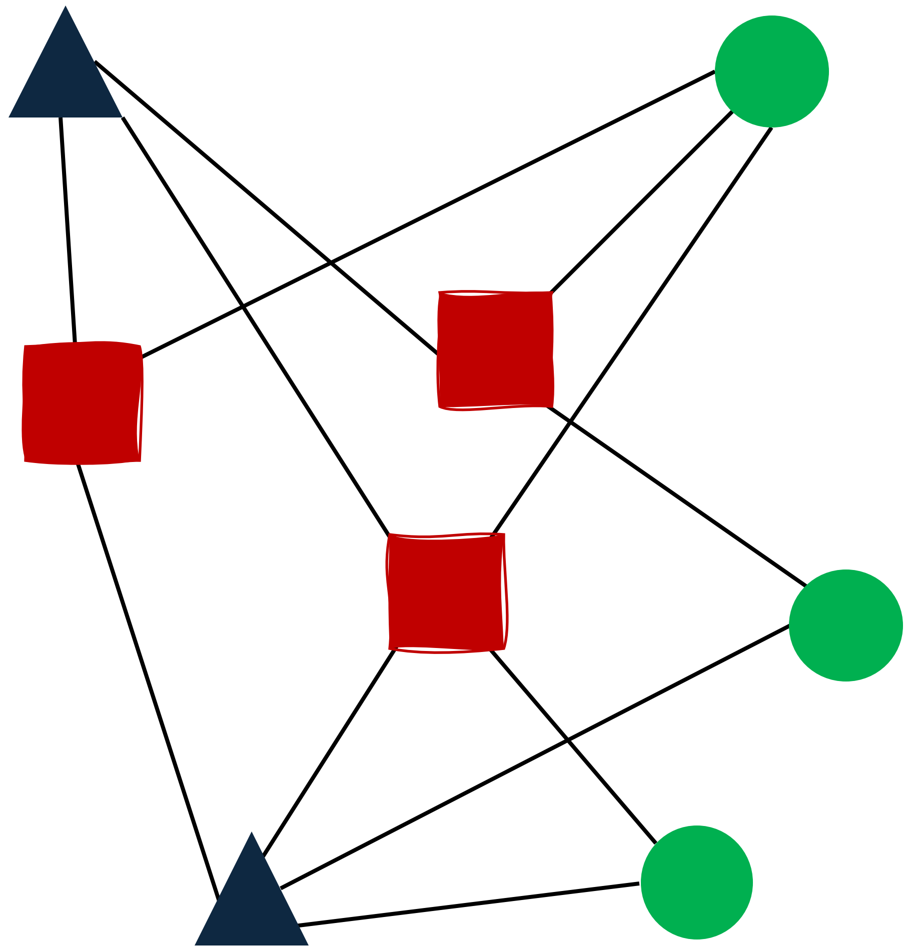


K-coloring

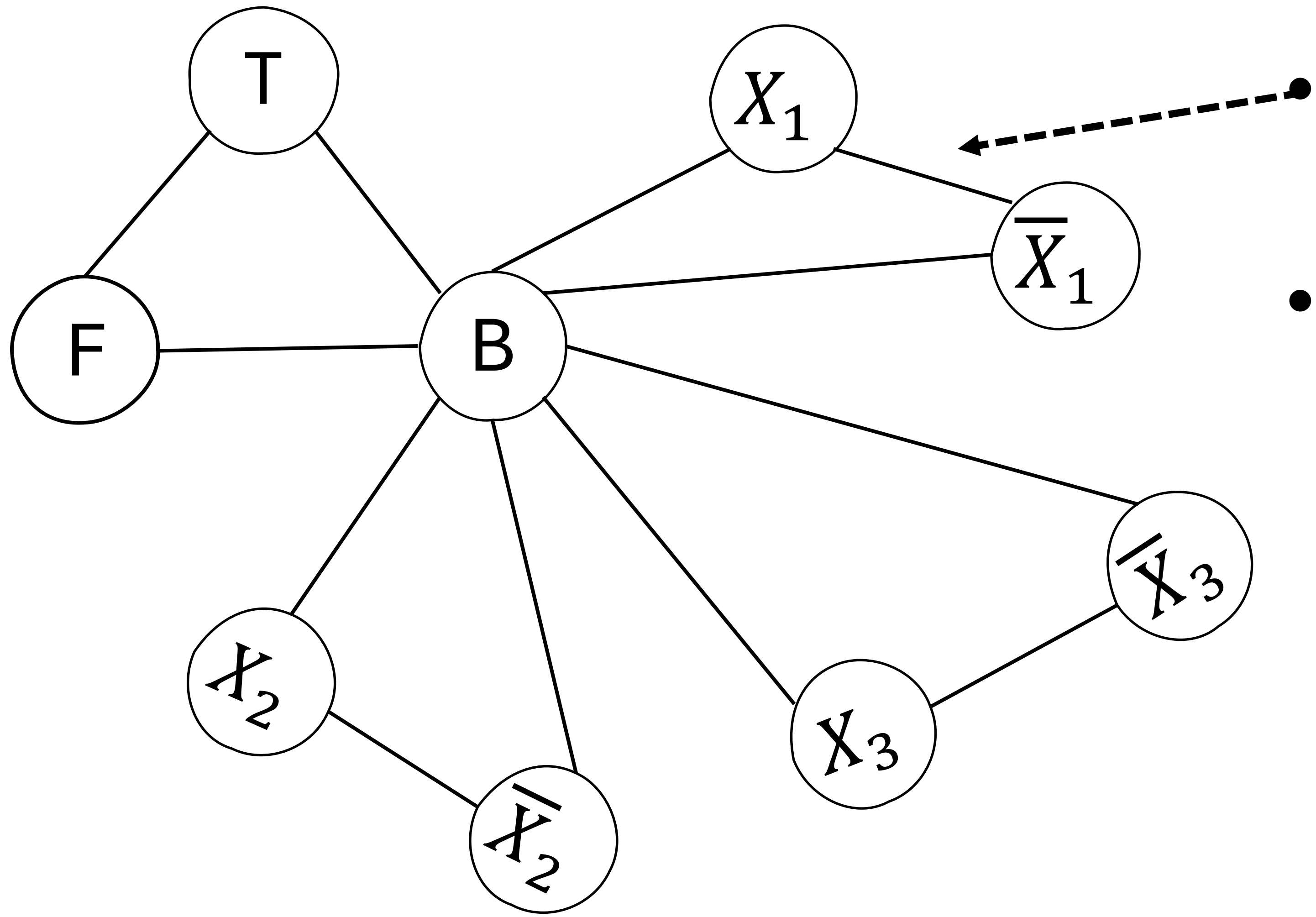
K-Coloring

- **Input**: Graph G and integer k.
- **Output**: Does there exist a k-coloring of G.

This in NP-hard because you can reduce from 3-SAT.



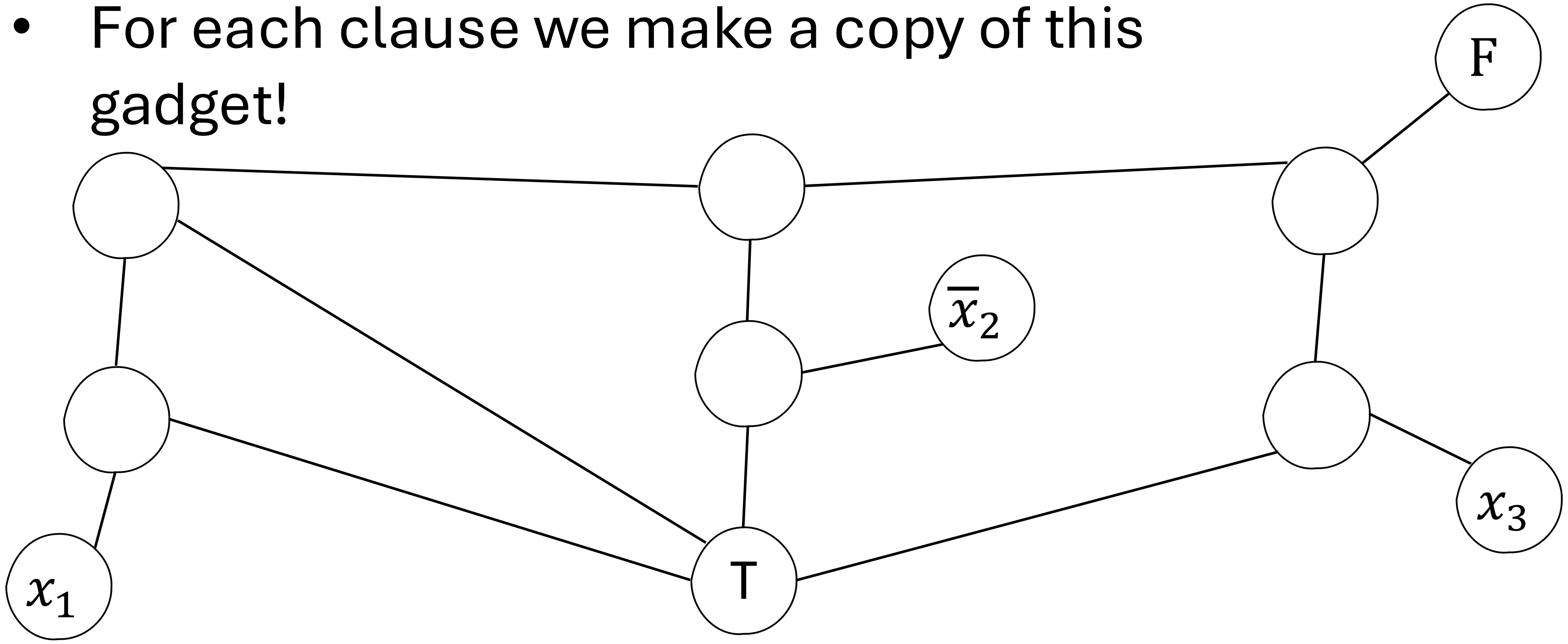
3-SAT $\leq_p$ 3-coloring



- Create a triangle for each variable.
- Every vertex will be touching the B vertex and thus must only use on of the colors assigned to F or T.

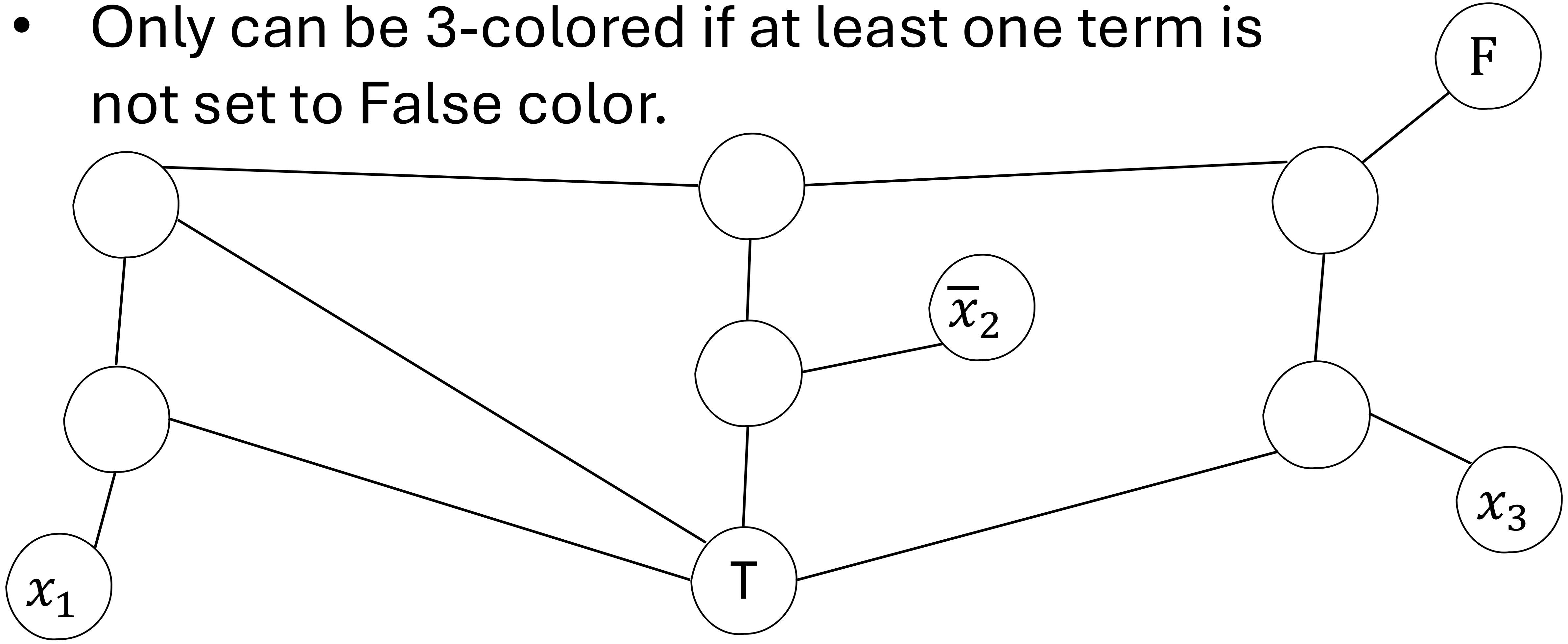
3-SAT $\leq_p$ 3-coloring

- For each clause we make a copy of this gadget!



3-SAT $\leq_p$ 3-coloring

- Only can be 3-colored if at least one term is not set to False color.



3-SAT $\leq_p$ 3-coloring

- Observe that we can construct the gadgets in O(m) time.
- If there exists a 3-coloring then each variable will get an assignment in its triangle (it will match the color of T or F nodes).
- If there is a satisfying assignment, then we can make all terms that evaluate to true Blue (1), all terms evaluate toe false Red (2), and everything else Green (3).

CSE 331: Algorithms & Complexity

Thank you

University at Buffalo