Lecture 25

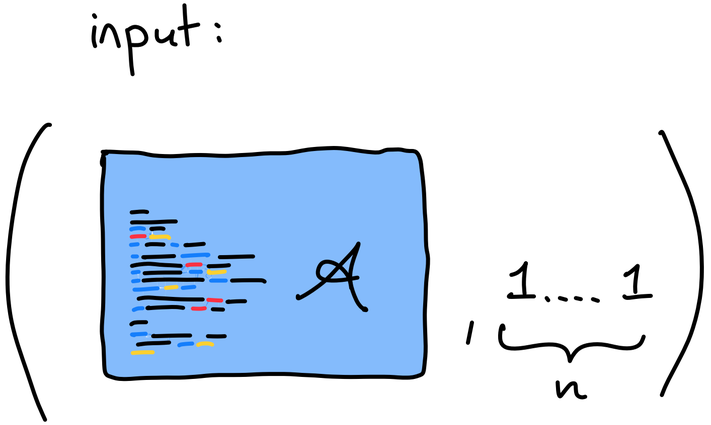
NP completeness III

Chinmay Nirkhe | CSE 421 Spring 2025

Previously in CSE 421…

The “first” NP-complete problem
Satisfiability

- **Satisfiability:** Input: $(\langle \mathscr{A} \rangle, n)$, the description of an algorithm $\mathscr{A}$ and integer n in unary. Output: Whether there exists a π such that $\mathscr{A}(\pi) = 1$ and $|\pi| = n$.
- **Theorem:** Satisfiability is NP-complete.
- **Proof:**
  - Satisfiability is in NP as π is a proof of the satisfiability.
  - For any other problem $X \in$ NP, there exists a certifier $\mathscr{V}(x, \pi)$ such that x is a “yes” instance iff there exists a π such that $\mathscr{V}(x, \pi)$ accepts.
    - Let $n = |\pi|$ taken as input by $\mathscr{V}$.
    - Define $\mathscr{A}(\pi) :=$ as the poly-sized program computing $\mathscr{V}(x, \pi)$ for x “hardcoded”.
    - Then x is a “yes” instance iff exists a π such that $\mathscr{A}(\pi) = 1$ and $|\pi| = n$.
    - So $X \leq_p Y$, proving NP-completeness.



Today

Proving more NP-complete problems

- **Recipe** for showing that problem Y is NP-complete
  - Step 1: Show that $Y \in$ NP.
  - Step 2: Choose a known NP-compete problem X.
  - Step 3: Prove that $X \leq_p Y$.
- **Correctness** of recipe: We claim that $\leq_p$ is a transitive operation.
  - If $W \leq_p X$ and $X \leq_p Y$ then $W \leq_p Y$.
  - For any problem $W \in$ NP, then $W \leq_p Y$, proving that Y is NP-complete.

3-SAT problem

- The 3-SAT problem is the most well known of all NP-complete problems
- A boolean formula φ is a 3-SAT formula over variables $x_1, \ldots, x_n \in \{0,1\}$ if
  - $\varphi = \varphi_1 \wedge \varphi_2 \wedge \ldots \wedge \varphi_k$, the "AND" of k-subformulas
  - Each φ_j is the "OR" of ≤ 3 variables or their negations from $x_1, \ldots, x_n$.
- Examples: $\varphi(z_1, \ldots, z_4) = (z_1 \vee \neg z_2 \vee z_3) \wedge (\neg z_1 \vee \neg z_2 \vee z_4)$
- **Theorem:** 3-SAT is NP-complete.

Proof that 3-SAT is NP-complete

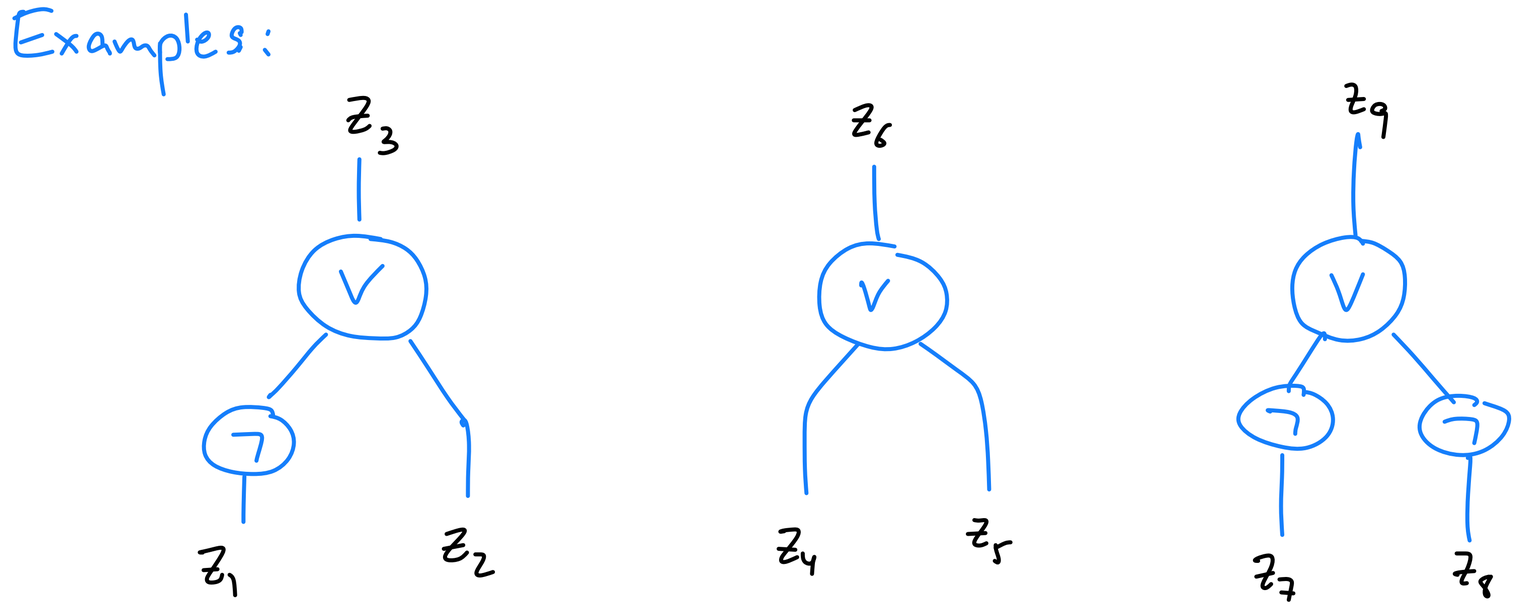
- **Key idea:** Show that Satisfiability reduces to 3-SAT.
- **Proof:** We saw that 3-SAT is in NP (the proof is just the satisfying assignment).
  - To show that Satisfiability reduces to 3-SAT, we follow the following outline to convert an instance of Satisfiability into an instance of 3-SAT:
  - Step 1: Convert every input $(\langle \mathscr{A} \rangle, n)$ to Satisfiability into a boolean circuit G.
  - Step 2: Adjust G so that it is nicely structured: Use De Morgan's laws to ensure G consists of only OR and NOT gates, has no double negations.
  - Step 3: Label every input wire and output wire of an OR gate, with a variable z_i.
  - Step 4: Convert each gate of G into a set of clauses in the 3-SAT formula.
  - Step 5: The circuit must output the value 1 (so that it's a yes instance)

Proof that 3-SAT is NP-complete

- Step 2 elaborated:
  - De Morgan's laws:
    - Switching ANDs to ORs: $(y_1 \wedge y_2) = \neg(\neg y_1 \vee \neg y_2)$
    - Double negations: $\neg\neg y_1 = y_1$
    - Decomposing Big ORs: $y_1 \vee y_2 \vee y_3 \vee y_4 = (y_1 \vee y_2) \vee (y_3 \vee y_4)$
  - Using these boolean formula transforms, any boolean circuit can be converted into one with only OR and NOT gates

Proof that 3-SAT is NP-complete

- Step 3: Label every input wire and output wire of an OR gate, with a variable z_i.

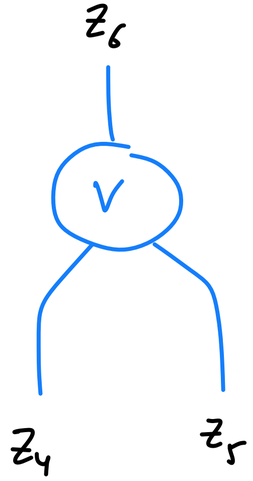


Proof that 3-SAT is NP-complete

Remember that $a \Rightarrow b$ is equiv. to $\neg a \vee b$.

- Step 4: Convert each gate of G into clauses to include in the 3-SAT formula.

Consider the following OR gate.



We want $z_6 \iff z_4 \vee z_5$.

$$= \big((z_4 \vee z_5) \Rightarrow z_6\big) \wedge \big(z_6 \Rightarrow (z_4 \vee z_5)\big)$$

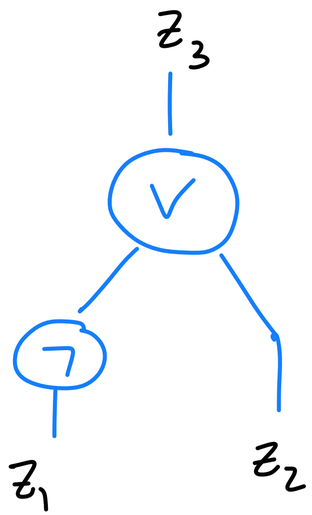
$$= \big(z_4 \Rightarrow z_6\big) \wedge \big(z_5 \Rightarrow z_6\big) \wedge \big(z_6 \Rightarrow (z_4 \vee z_5)\big)$$

$$= \big(\neg z_4 \vee z_6\big) \wedge \big(\neg z_5 \vee z_6\big) \wedge \big(\neg z_6 \vee z_4 \vee z_5\big)$$

Proof that 3-SAT is NP-complete

- Step 4: Convert each gate of G clauses to include in the 3-SAT formula.

Consider the following OR gate.



Construct the following clauses:

$$(\neg(\neg z_1) \vee z_3) \wedge (\neg z_2 \vee z_3) \wedge (\neg z_3 \vee (\neg z_1) \vee z_2)$$

$$= (z_1 \vee z_3) \wedge (\neg z_2 \vee z_3) \wedge (\neg z_3 \vee \neg z_1 \vee z_2)$$

Proof that 3-SAT is NP-complete

- Step 4: Convert each gate of G into clauses to include in the 3-SAT formula.
- **Key lemma:** If a 3-CNF φ includes $(\neg a \vee c)$, $(\neg b \vee c)$, $(\neg c \vee a \vee b)$ as clauses, then any satisfying assignment for the φ must set c to equal $a \vee b$.
- **Proof**:
  - The clauses imply the following statements: $a \Rightarrow c$, $b \Rightarrow c$, $c \Rightarrow (a \vee b)$.
  - The first two combine to $(a \vee b) \Rightarrow c$.
  - Therefore, $c \Leftrightarrow (a \vee b)$.

Proof that 3-SAT is NP-complete

- Step 5: The circuit must output the value 1 (so that it's a yes instance)
- **Solution:** Add a clause z_f where this is the variable corresponding to the final wire in the circuit
  - When the circuit has a satisfying input, there is an assignment of values to wires such that z_f is assigned to be 1
  - When the circuit has no satisfying input, if all other wires are consistently assigned, z_f is always assigned to be 0

Proof that 3-SAT is NP-complete

- Proving that the reduction is correct:
  - The reduction is polynomial time as it takes a constant number of passes over the input to generate (we don't need any answer more specific than this).
  - "Yes" $\rightarrow$ "Yes": If $(\langle \mathscr{A} \rangle, n)$ is a "Yes" instance, then there is an input x of length n such that $\mathscr{A}(x) = 1$. Let z be the value of the wires of the corresponding circuit G. Then z satisfies the 3-SAT φ by construction.
  - "Yes" $\leftarrow$ "Yes": If z satisfies the 3-SAT φ, then let x be the values assigned by z to the inputs. Then $G(x) = 1$ as each intermediate gate will evaluate to match z due to the previous lemma. And $\mathscr{A}(x) = 1$ iff $G(x) = 1$ so $\mathscr{A}$ is satisfiable.

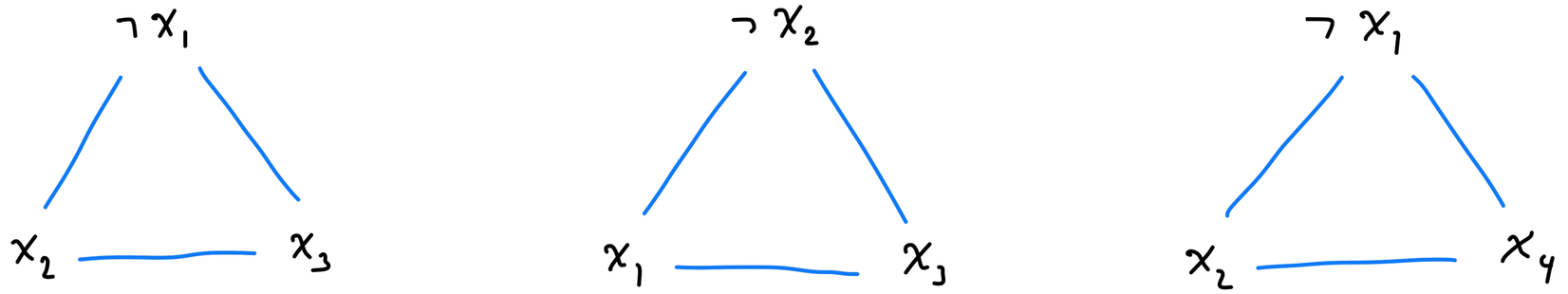
General suggestions about proving NP-completeness

- There is not a clear cut set of techniques you can always apply
- Proving NP-completeness is a bit of an art —
  - To prove problem Y is NP-complete, the most difficult step is finding a problem X which is known to be is NP-complete such that $X \leq_p Y$
  - You are converting instances of X into instances of Y
  - I.e. every instance of X is a special case of an instance of Y

Vertex Cover is NP-complete

- We've seen that Vertex Cover is in NP.
- Let's show that 3-SAT $\leq_p$ Vertex Cover.
- We need to create a graph G and integer k which captures the structure of a 3-SAT formula φ.
- **Construction:** G contains 3 vertices per clause, one per literal. $k = 2m$ where m is the number of clauses in φ.

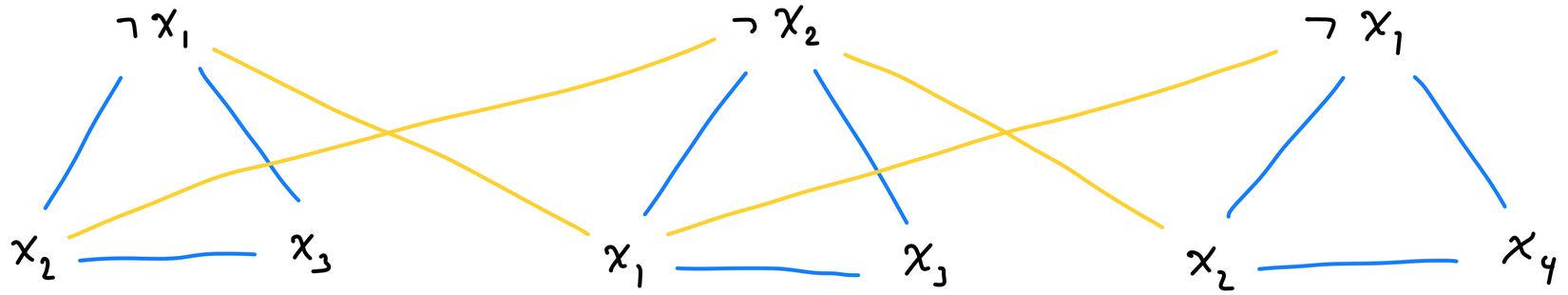
$$\varphi = (\neg x_1 \vee x_2 \vee x_3) \wedge (x_1 \vee \neg x_2 \vee x_3) \wedge (\neg x_1 \vee x_2 \vee x_4)$$



Vertex Cover is NP-complete

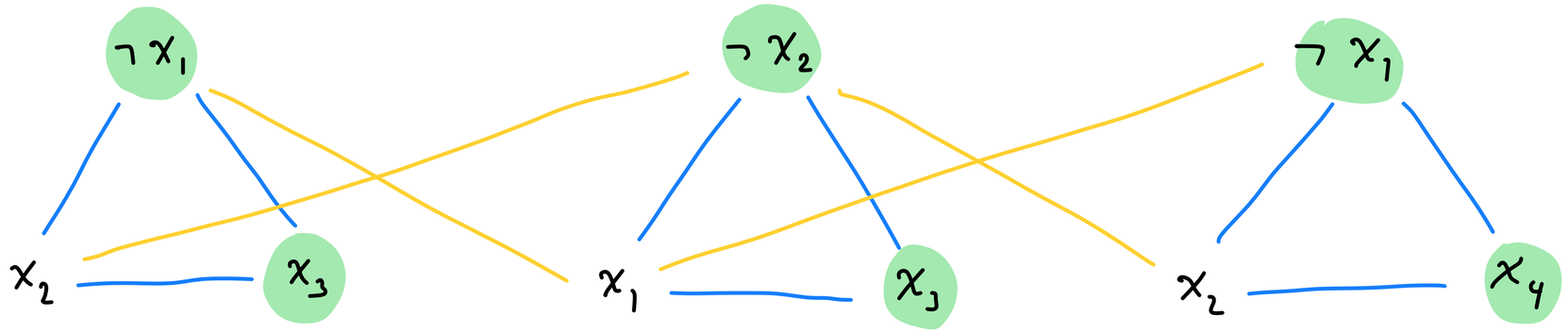
- We've seen that Vertex Cover is in NP.
- Let's show that 3-SAT $\leq_p$ Vertex Cover.
- We need to create a graph G and integer k which captures the structure of a 3-SAT formula φ.
- **Construction**:
  - G contains 3 vertices per clause, one per literal. $k = 2m$ where m is the number of clauses in φ.
  - Add an edge between each pair of literals in a clause. Add edges connecting each variable to its negation.

$$\varphi = (\neg x_1 \vee x_2 \vee x_3) \wedge (x_1 \vee \neg x_2 \vee x_3) \wedge (\neg x_1 \vee x_2 \vee x_4)$$



Vertex Cover is NP-complete

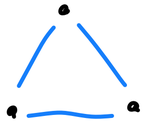
$$\varphi = (\neg x_1 \vee x_2 \vee x_3) \wedge (x_1 \vee \neg x_2 \vee x_3) \wedge (\neg x_1 \vee x_2 \vee x_4)$$

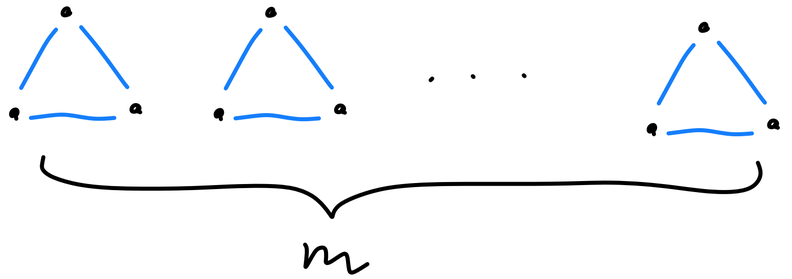


Observe: $x_1 = x_2 = 1$, $x_3 = x_4 = 0$ is a satisfying instance.

And the identified vertices form a vertex cover.

Vertex Cover is NP-complete

Observation: If a  exists in the graph, then at least 2 of the vertices must be included in a vertex cover.

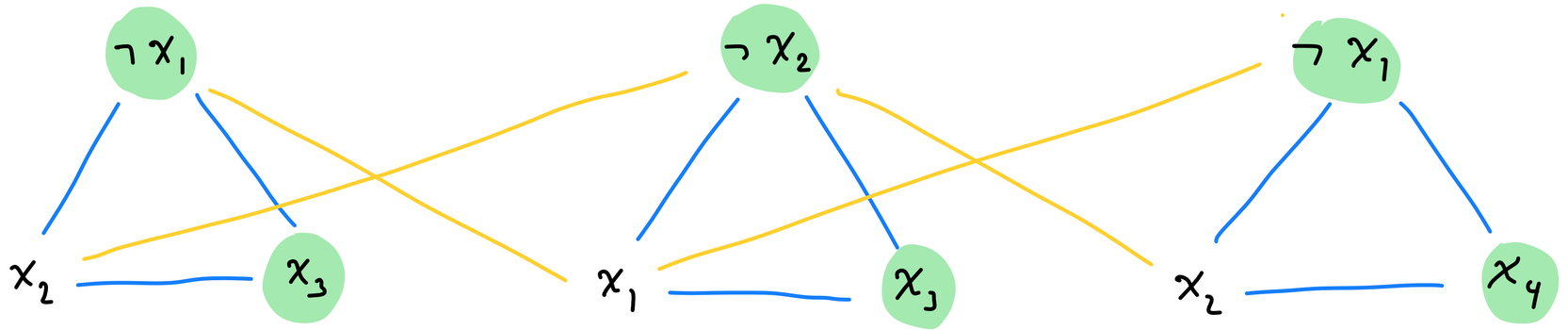
Corollary: If a graph has m disjoint triangles 

then the vertex cover must be size $\geq 2m$.

Vertex Cover is NP-complete

- **Theorem:** φ is satisfiable iff G has a vertex cover of size $\leq 2m$.
- **Proof**:
  - "Yes" $\rightarrow$ "Yes": If φ is satisfiable, let x be a satisfying assignment.
    - For each clause, pick one of the literals that must be set to be true and include the other two in the vertex cover. This is a vertex cover of size $2m$.
    - Each "triangle edge" is covered as 2 vertices are selected per triangle.
    - Each "negation edge" is covered as not selecting both endpoints would have both x_i and $\neg x_i$ to be true in the satisfying assignment.

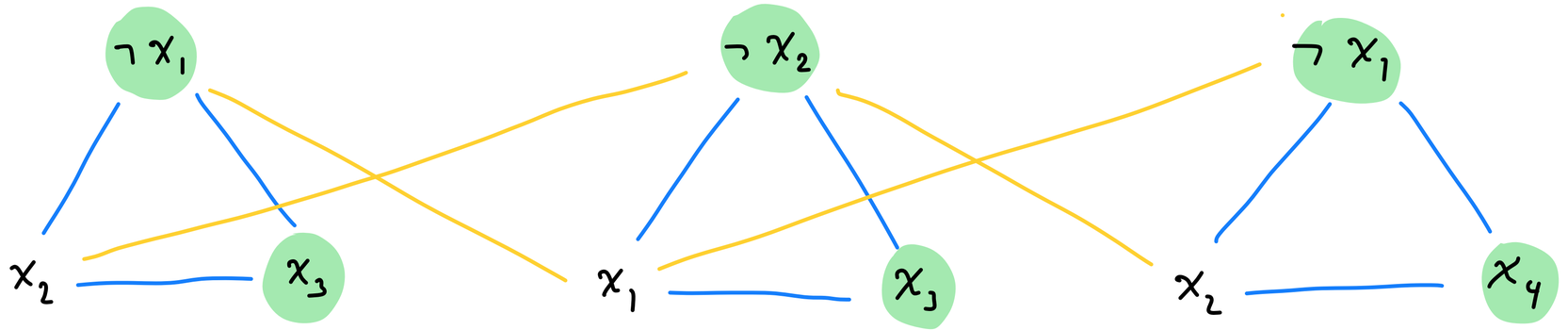
$$\varphi = (\neg x_1 \vee x_2 \vee x_3) \wedge (x_1 \vee \neg x_2 \vee x_3) \wedge (\neg x_1 \vee x_2 \vee x_4)$$



Vertex Cover is NP-complete

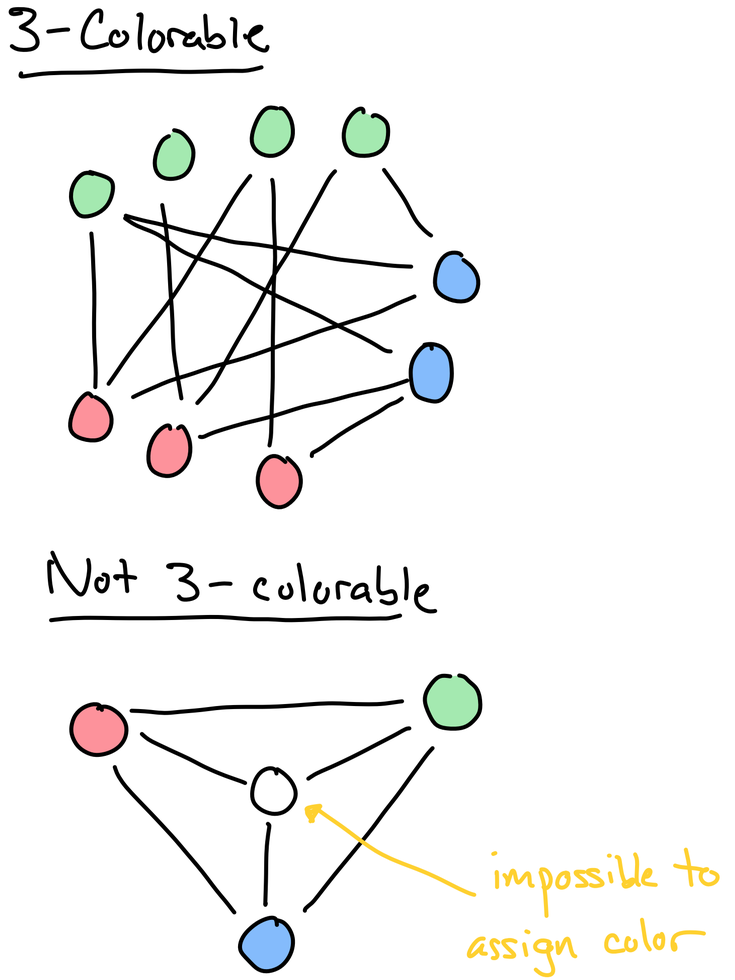
- **Theorem:** φ is satisfiable iff G has a vertex cover of size $\leq 2m$.
- **Proof**:
  - "Yes" $\leftarrow$ "Yes": If G has a vertex cover of size $\leq 2m$, then by previous lemma, the vertex cover is exactly size $2m$ and selects two vertices per triangle.
    - Set the excluded literal in each triangle to be *true*.
    - Since each "negation edge" is covered, the assignment will set at most one of x_i and $\neg x_i$ to be true.
    - Each clause is satisfied as one literate must be excluded in each clause.

$$\varphi = (\neg x_1 \vee x_2 \vee x_3) \wedge (x_1 \vee \neg x_2 \vee x_3) \wedge (\neg x_1 \vee x_2 \vee x_4)$$



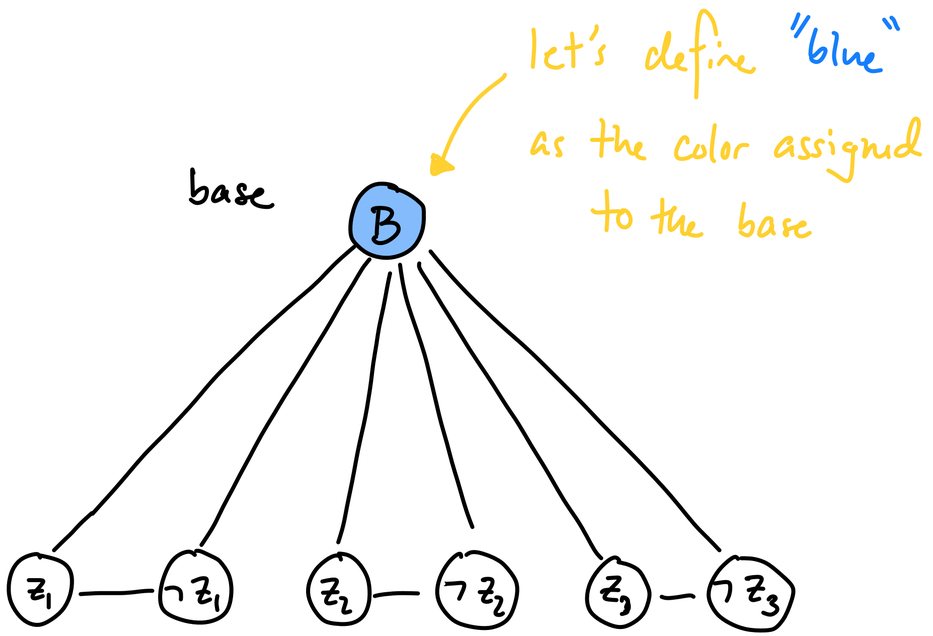
3-color is NP-complete

- **Input:** a graph $G = (V, E)$. **Output:** If there exists an assignment $\pi : V \to \{R, G, B\}$ such that $\pi(u) \neq \pi(v)$ for every edge $(u, v) \in E$
- 3-Color $\in$ NP as the proof is the assignment π
- We will show that 3-SAT $\leq_p$ 3-Color
  - We have to create a graph G representing a formula φ
  - Some “part” of the graph will have to represent variables and their negations
  - Some “part” of the graph will have to represent clauses such that the “part” can only be assigned colors if the clause is true

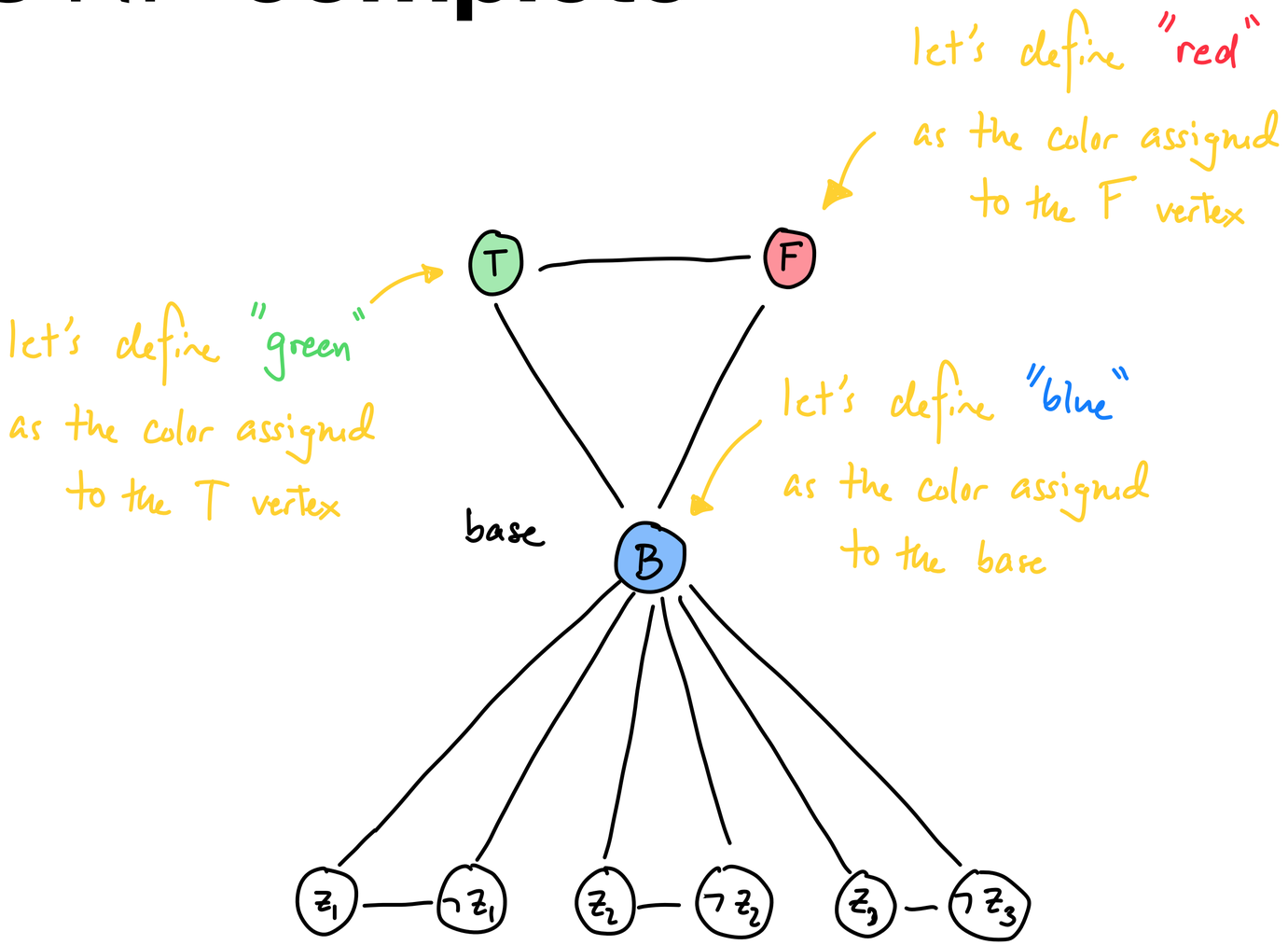


3-color is NP-complete

- For every variable z_i create a vertex z_i and $\neg z_i$
- Let's build a reduction such that
  - if z_i is colored GREEN then z_i should be set to be true
  - If z_i is colored RED then z_i should be set to be false
- By connecting triangles $B, z_i, \neg z_i$ we enforce that exactly one of z_i and $\neg z_i$ will be colored GREEN and RED
- So far the set of satisfying colorings are in bijection with assignments of the variables to true or false



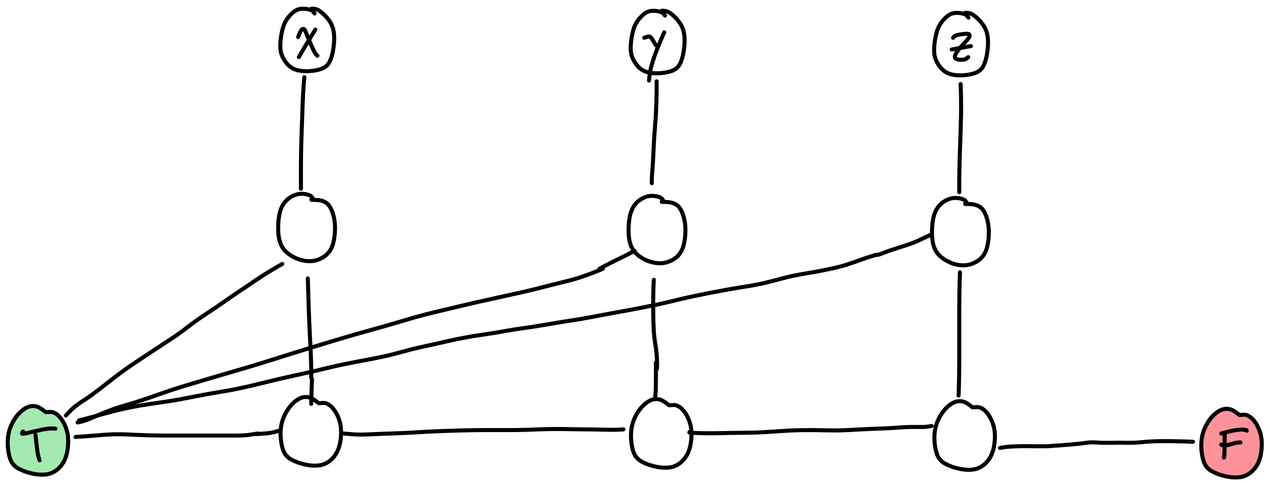
3-color is NP-complete



3-color is NP-complete

We now need to construct a "gadget" per clause $x \vee y \vee z$ s.t.

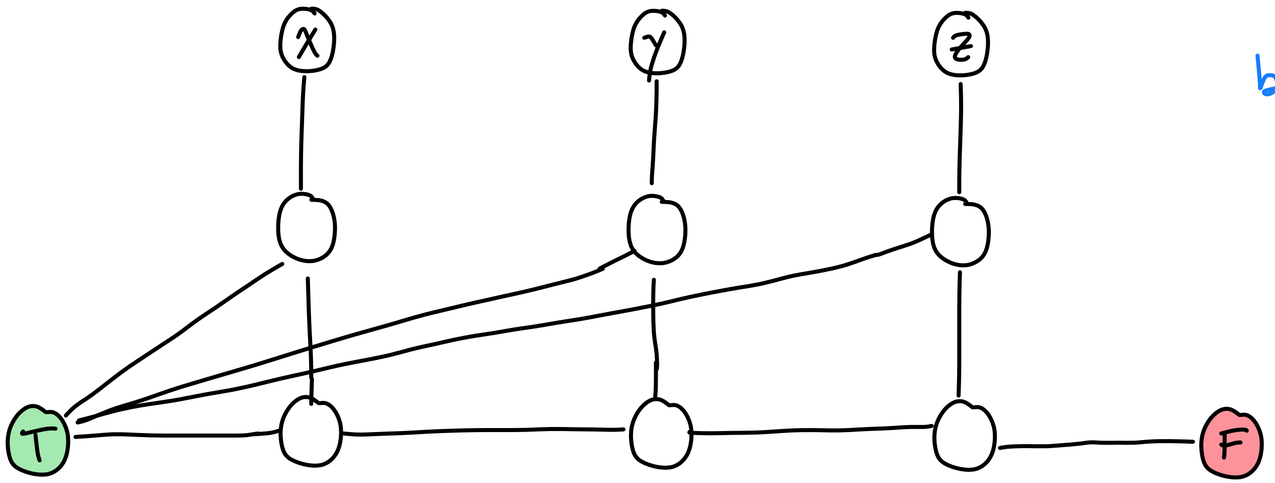
if all 3 corresponding vertices are colored red iff the gadget isn't colorable



3-color is NP-complete

We now need to construct a "gadget" per clause $x \vee y \vee z$ s.t.

if all 3 corresponding vertices are colored red iff the gadget isn't colorable

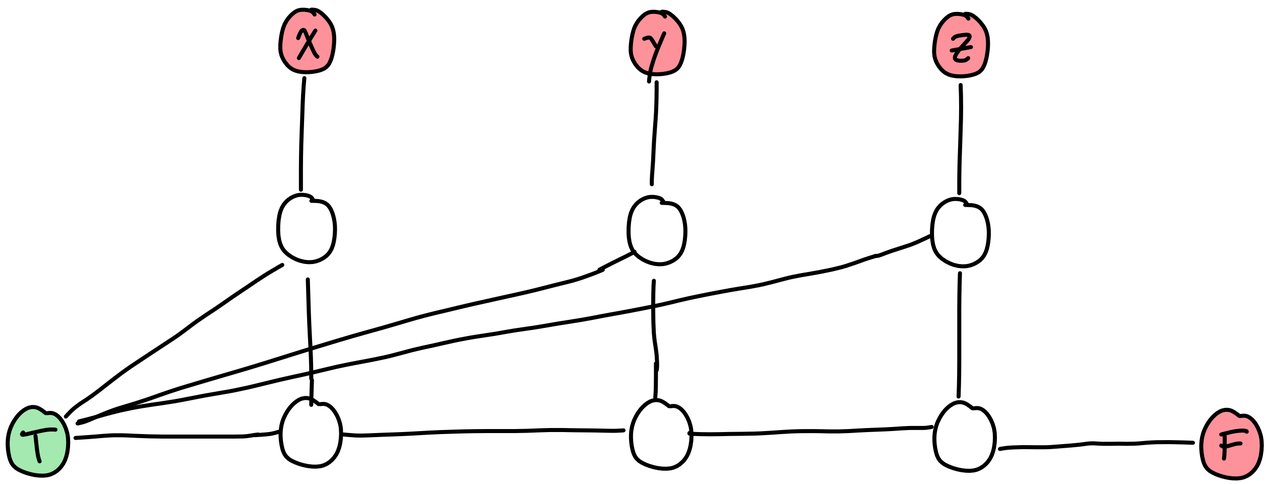


Recall every literal must be colored red or green by first construction.

3-color is NP-complete

We now need to construct a "gadget" per clause $x \vee y \vee z$ s.t.

if all 3 corresponding vertices are colored red iff the gadget isn't colorable

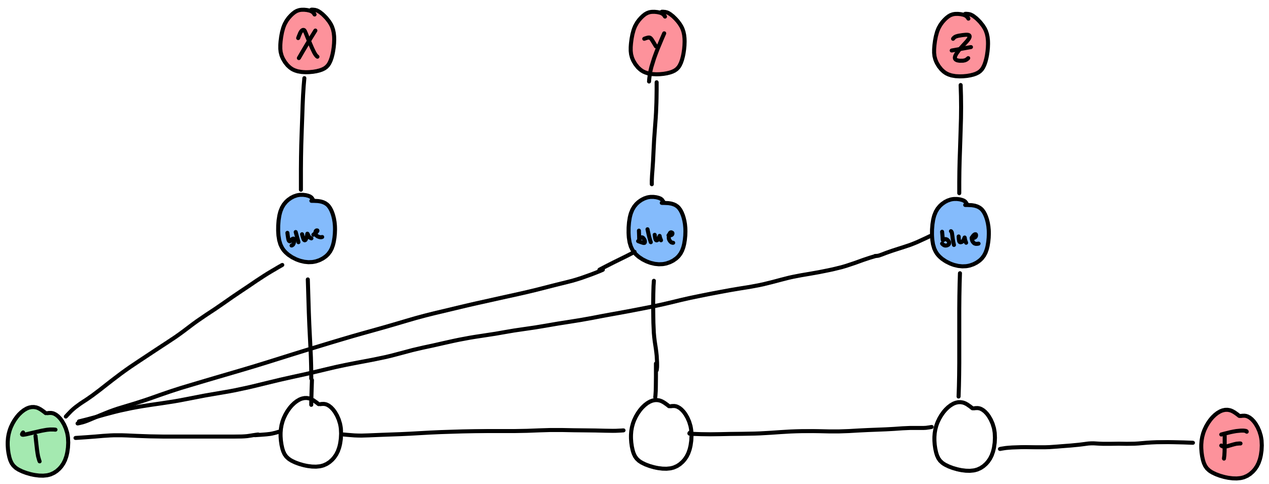


Case 1: x, y, z are all set to red

3-color is NP-complete

We now need to construct a "gadget" per clause $x \vee y \vee z$ s.t.

if all 3 corresponding vertices are colored red iff the gadget isn't colorable

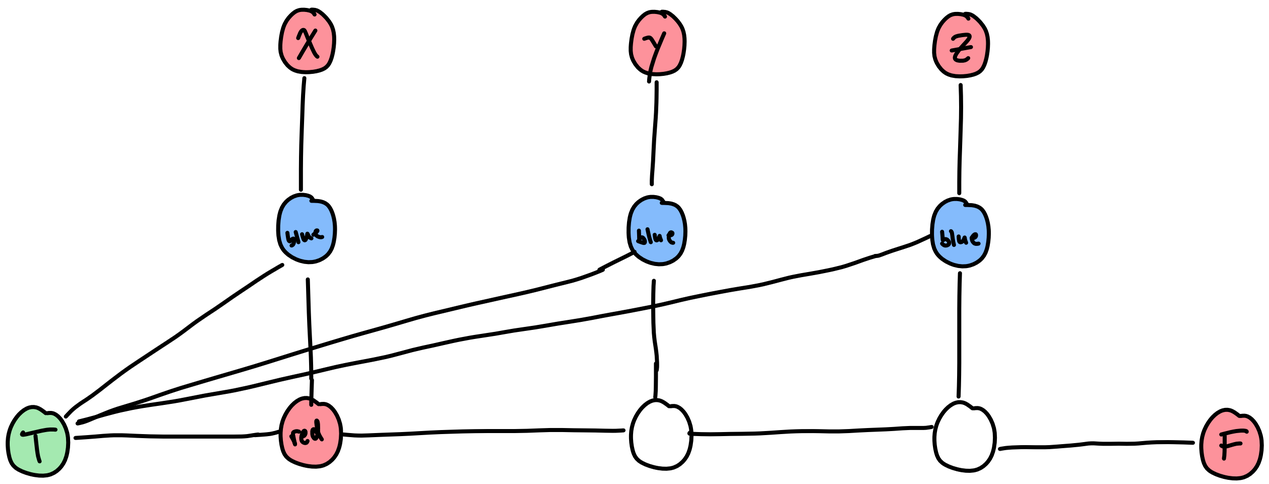


Case 1: x, y, z are all set to red

3-color is NP-complete

We now need to construct a "gadget" per clause $x \vee y \vee z$ s.t.

if all 3 corresponding vertices are colored red iff the gadget isn't colorable

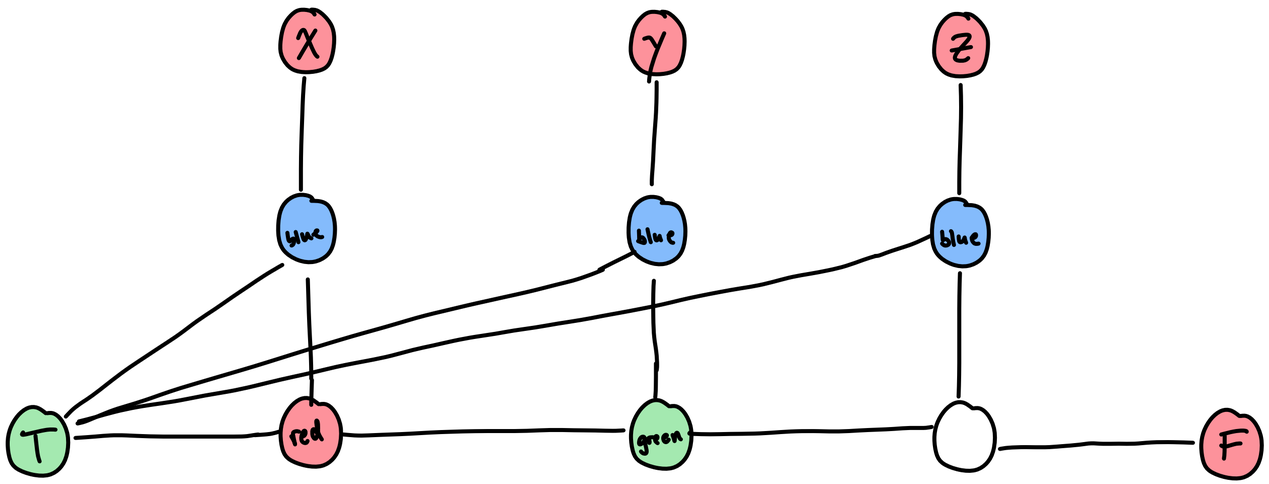


Case 1: x, y, z are all set to red

3-color is NP-complete

We now need to construct a "gadget" per clause $x \vee y \vee z$ s.t.

if all 3 corresponding vertices are colored red iff the gadget isn't colorable

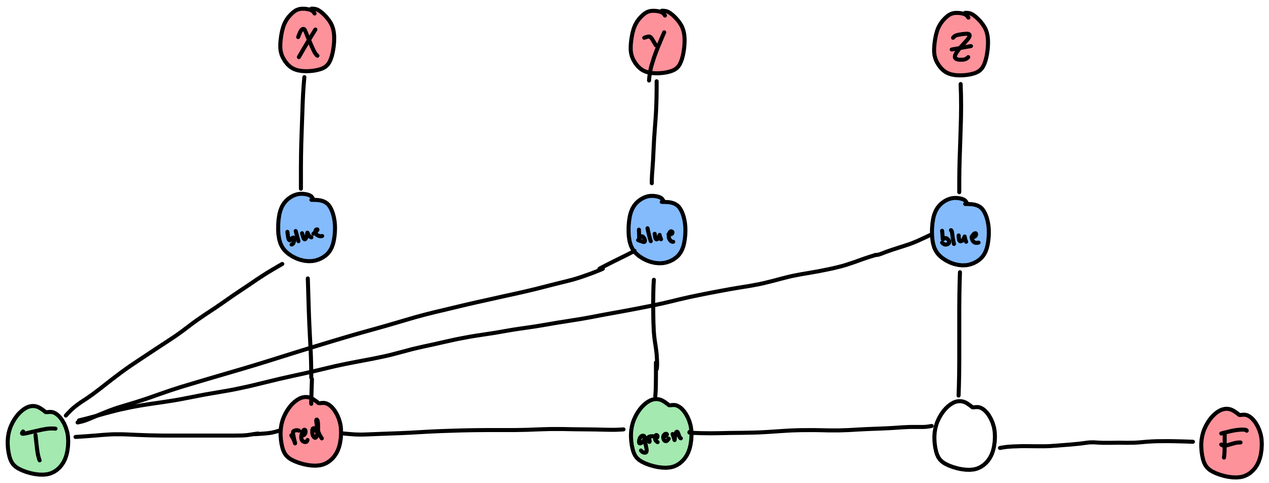


Case 1: x, y, z are all set to red

3-color is NP-complete

We now need to construct a "gadget" per clause $x \vee y \vee z$ s.t.

if all 3 corresponding vertices are colored red iff the gadget isn't colorable



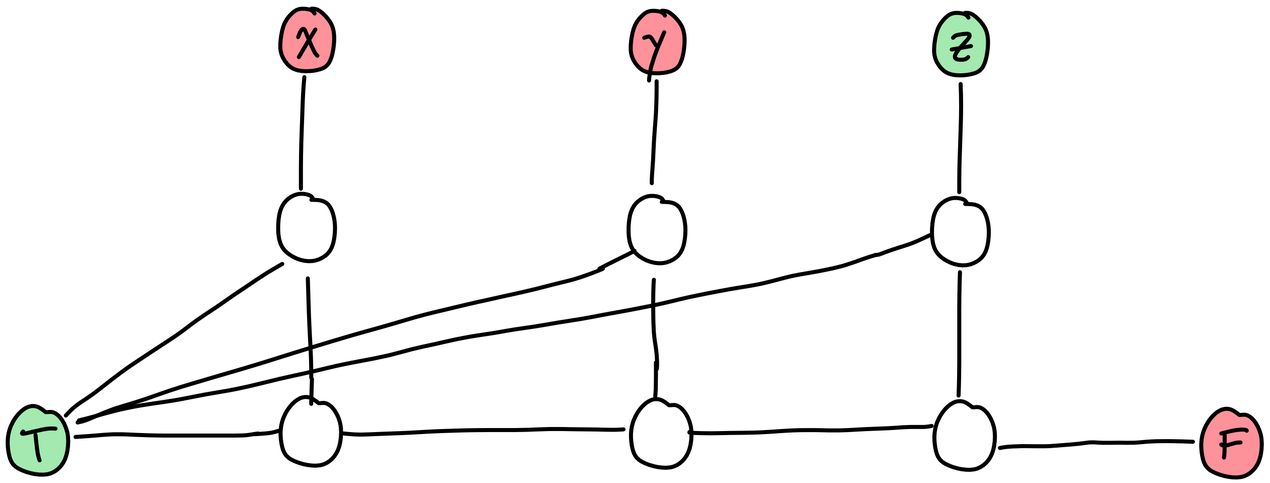
Case 1: x, y, z are all set to red

there is no color we can assign!

3-color is NP-complete

We now need to construct a "gadget" per clause $x \vee y \vee z$ s.t.

if all 3 corresponding vertices are colored red iff the gadget isn't colorable

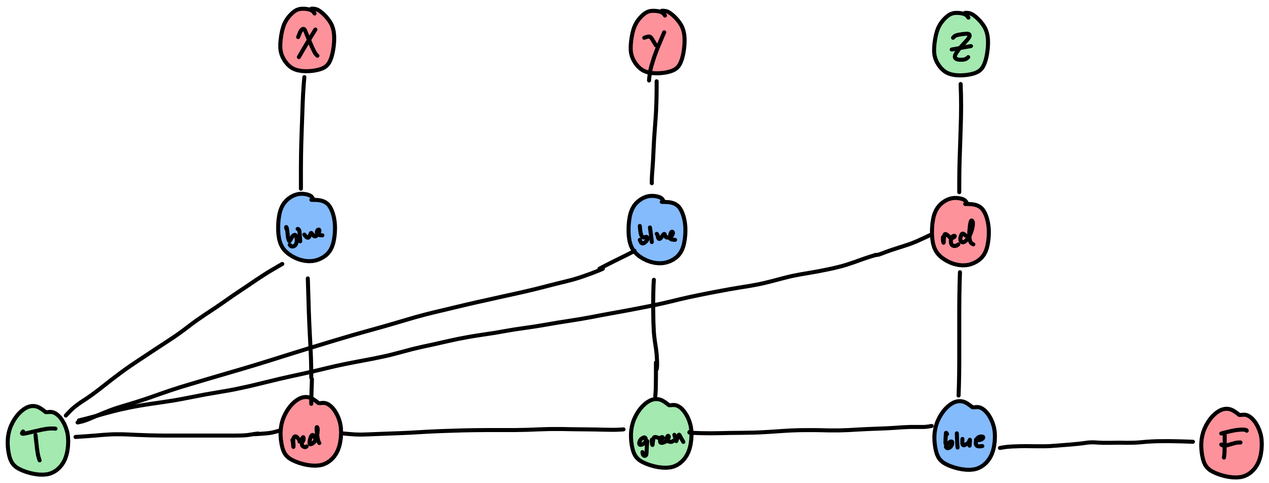


Case 2: x, y are colored red and z is colored green

3-color is NP-complete

We now need to construct a "gadget" per clause $x \vee y \vee z$ s.t.

if all 3 corresponding vertices are colored red iff the gadget isn't colorable

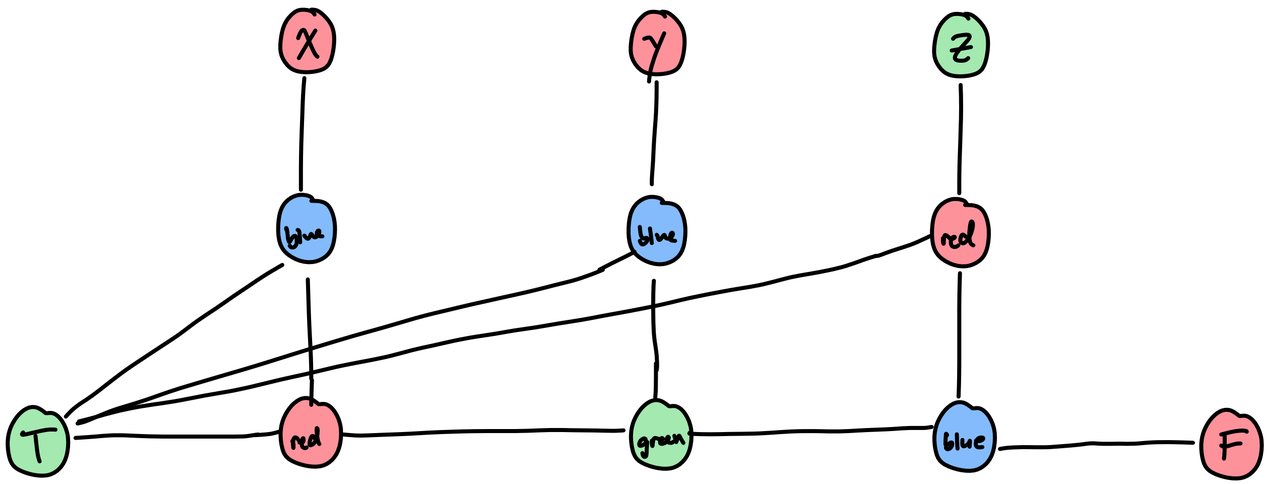


Case 2: x, y are colored red and z is colored green

3-color is NP-complete

We now need to construct a "gadget" per clause $x \vee y \vee z$ s.t.

if all 3 corresponding vertices are colored red iff the gadget isn't colorable

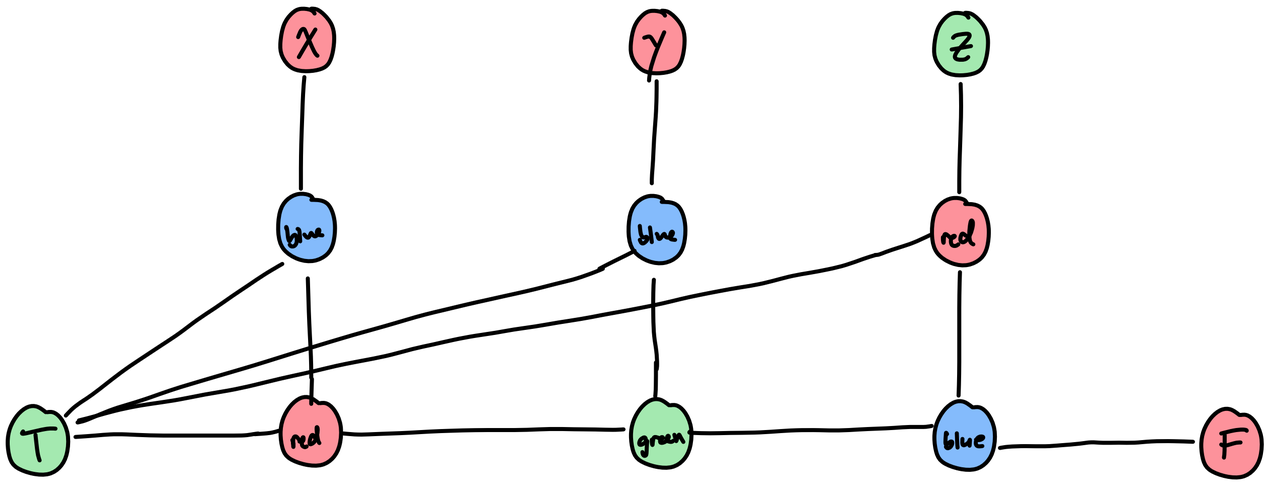


Case 2: x, y are colored red and z is colored green

3-color is NP-complete

We now need to construct a "gadget" per clause $x \vee y \vee z$ s.t.

if all 3 corresponding vertices are colored red iff the gadget isn't colorable

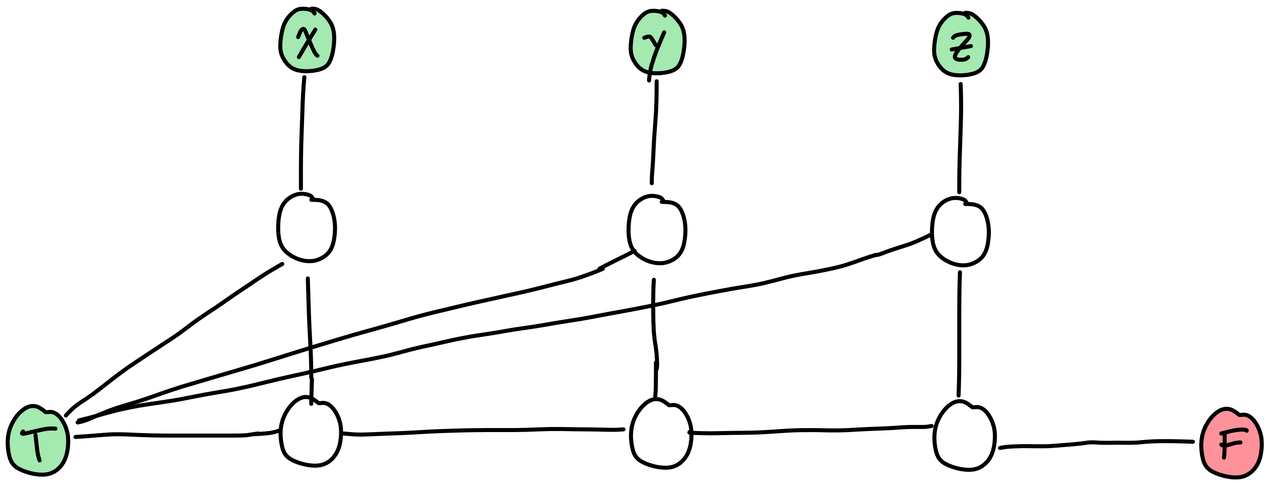


Case 2: x, y are colored red and z is colored green

3-color is NP-complete

We now need to construct a "gadget" per clause $x \vee y \vee z$ s.t.

if all 3 corresponding vertices are colored red iff the gadget isn't colorable

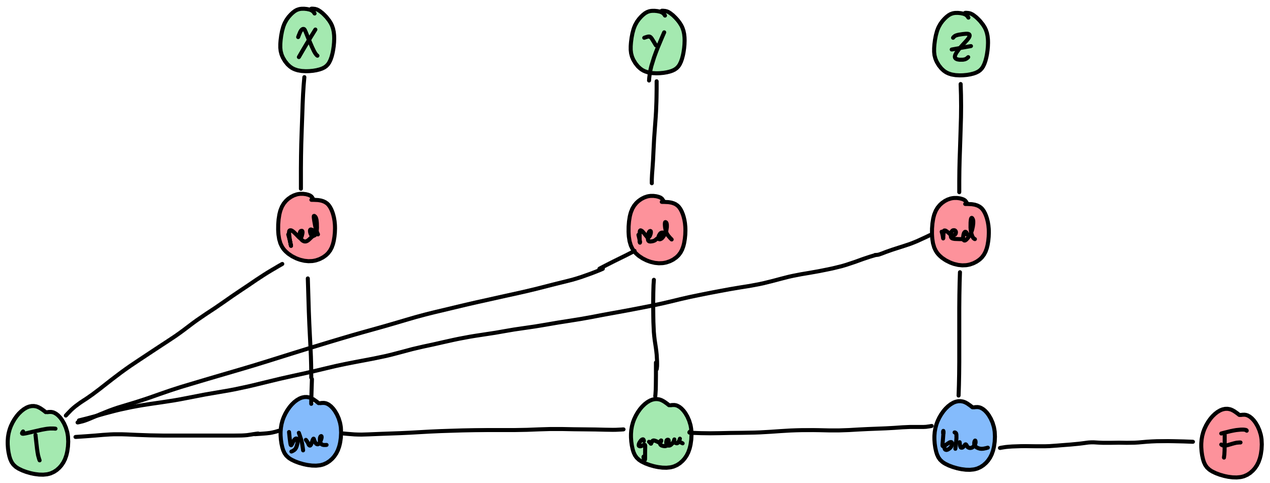


... Case 8: All 3 vertices x, y, z are colored green

3-color is NP-complete

We now need to construct a "gadget" per clause $x \vee y \vee z$ s.t.

if all 3 corresponding vertices are colored red iff the gadget isn't colorable



... Case 8: All 3 vertices x, y, z are colored green

3-color is NP-complete
Putting it all together

- Full construction:
  - Construct triangles (T,F,B) and (B, z_i, ¬z_i) for each variable z_i.
  - Construct gadget from vertices (x, y, z, T, F) as shown for each clause $x \vee y \vee z$
- Properties:
  - Every vertex on a triangle must have a different color in a valid coloring
  - Let GREEN be the color assigned to T, RED assigned to F, BLUE assigned to B
  - Lem: Exactly one of variable z_i and ¬z_i must be assigned GREEN or RED in a valid coloring
  - Lem: In a valid coloring, the gadget for $x \vee y \vee z$ is colorable iff one of x, y, z is colored GREEN

3-color is NP-complete
Putting it all together

- **Reduction proof:**
  - "Yes" $\rightarrow$ "Yes": Let z be a satisfying assignment to 3-SAT φ.
    - Color the vertices of z_i and $\neg z_i$ GREEN or RED respectively
    - Every clause is satisfied so there exists an assignment of colors for the gadget
  - "Yes" $\leftarrow$ "Yes": Let GREEN be the color assigned to T, RED assigned to F, BLUE assigned to B
    - Set z_i to be 1 if assigned color GREEN or 0 if assigned color RED
    - Since the gadget for clause $x \vee y \vee z$ has a valid coloring, at least one of the 3 literals must be GREEN and therefore the clause is satisfied