Lecture 23

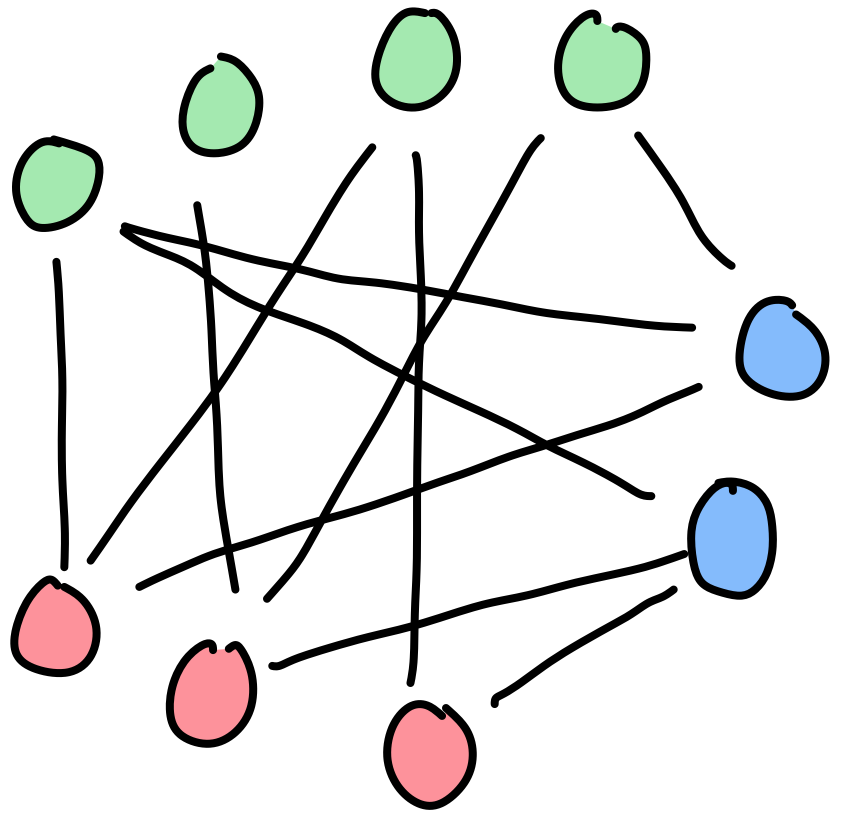
NP completeness IV

Chinmay Nirkhe | CSE 421 Winter 2026

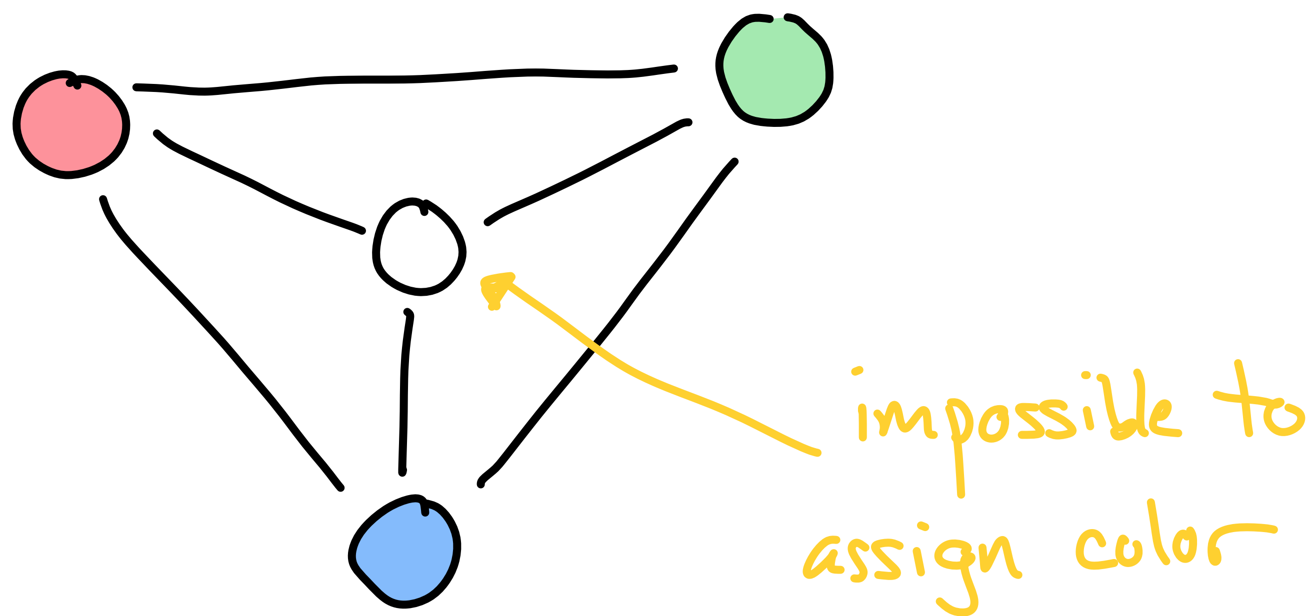
3-color is NP-complete

- **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-Colorable

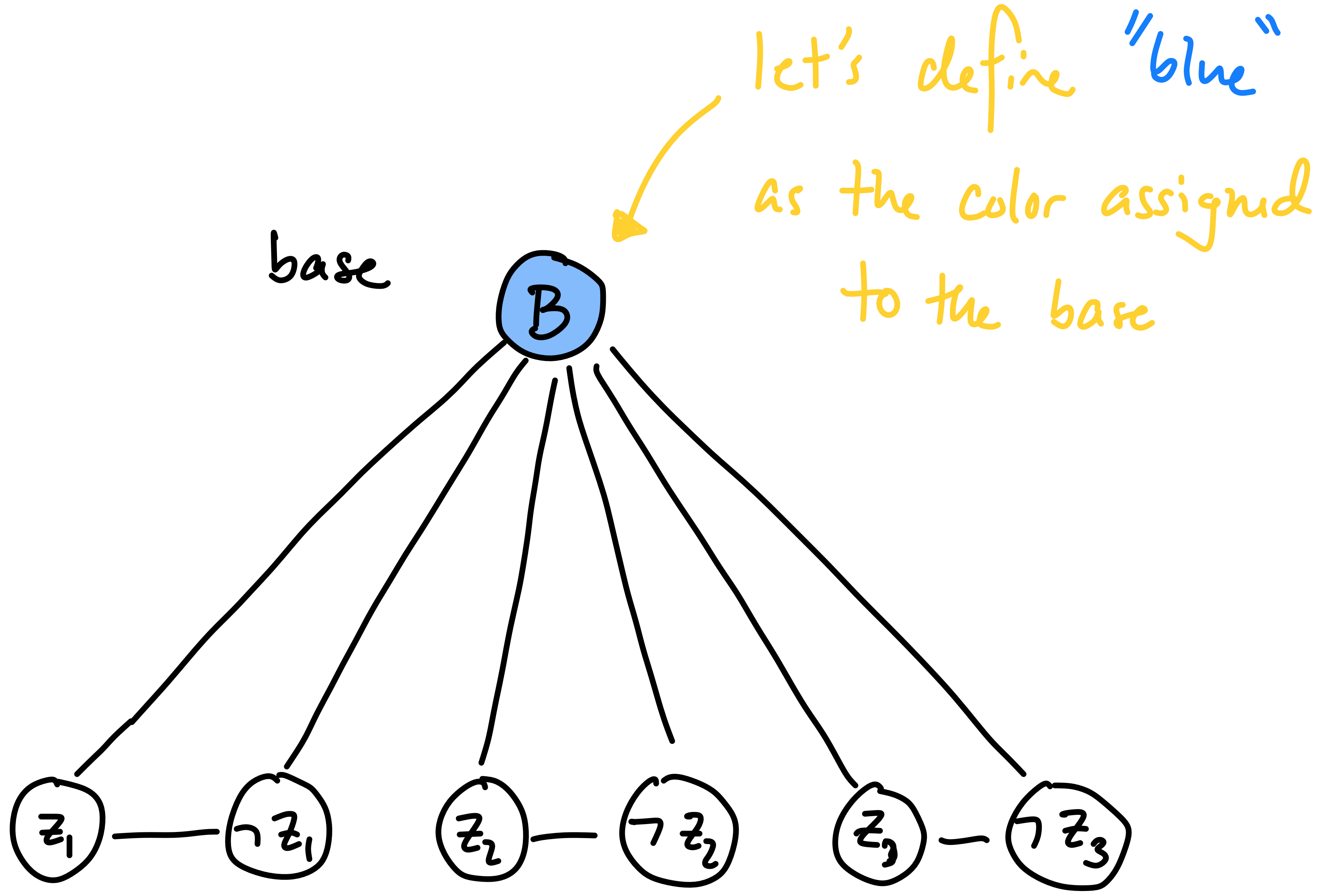


Not 3-colorable

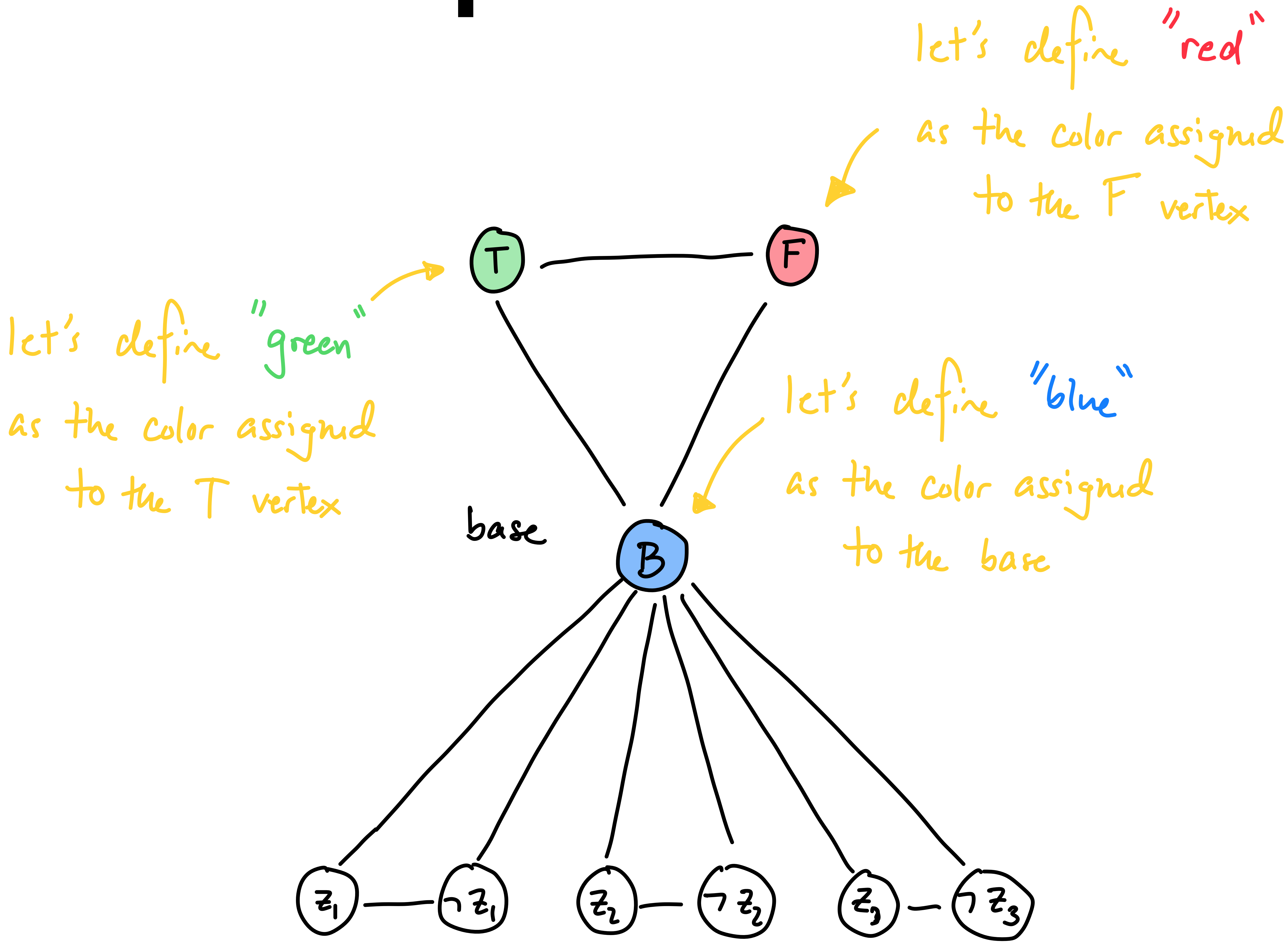


3-color is NP-complete

- 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



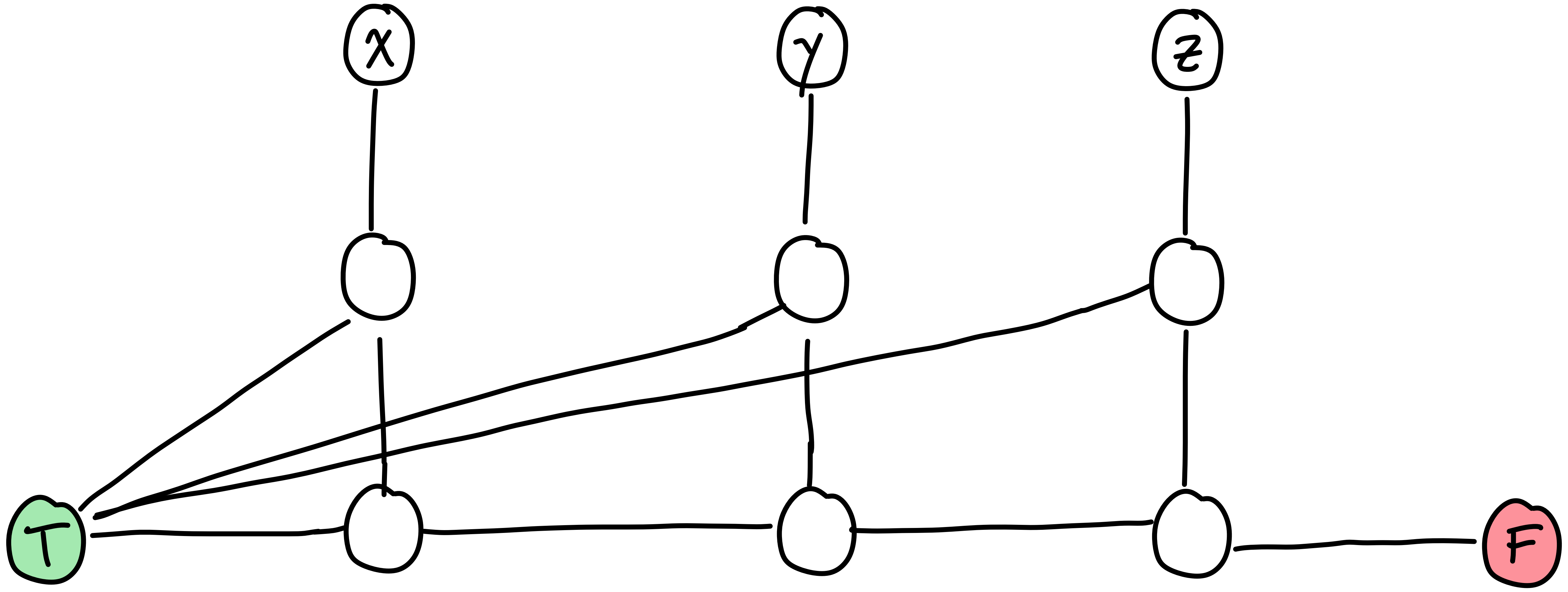
3-color is NP-complete



3-color is NP-complete

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

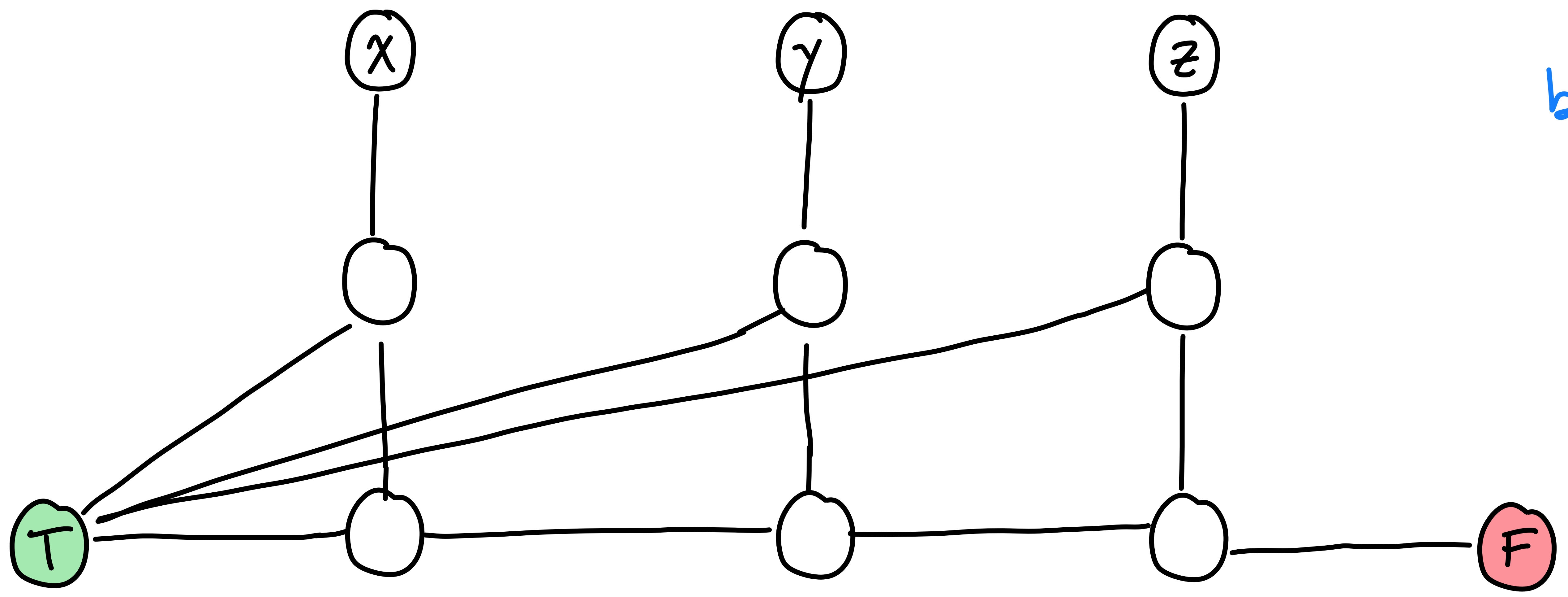
if all 3 corresponding vertices are colored red iff the gadget isn't colorable



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



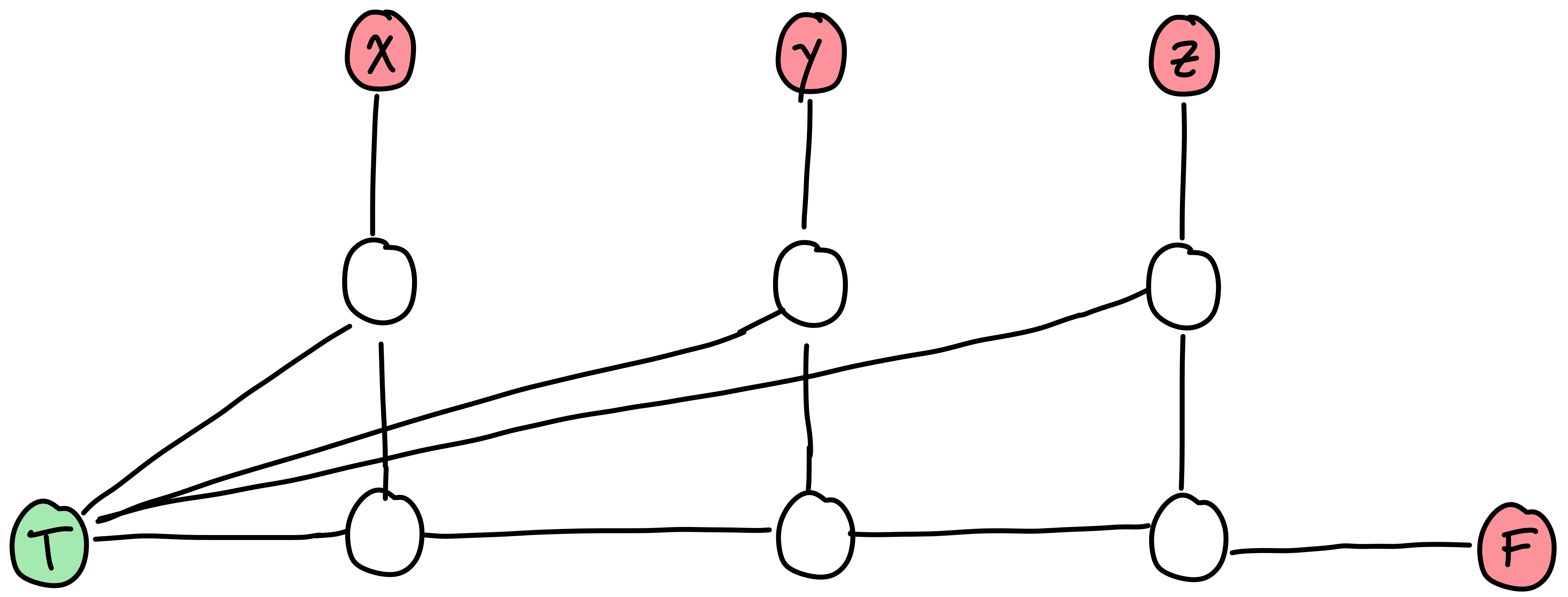
Recall every literal must be colored red or green by first construction.

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

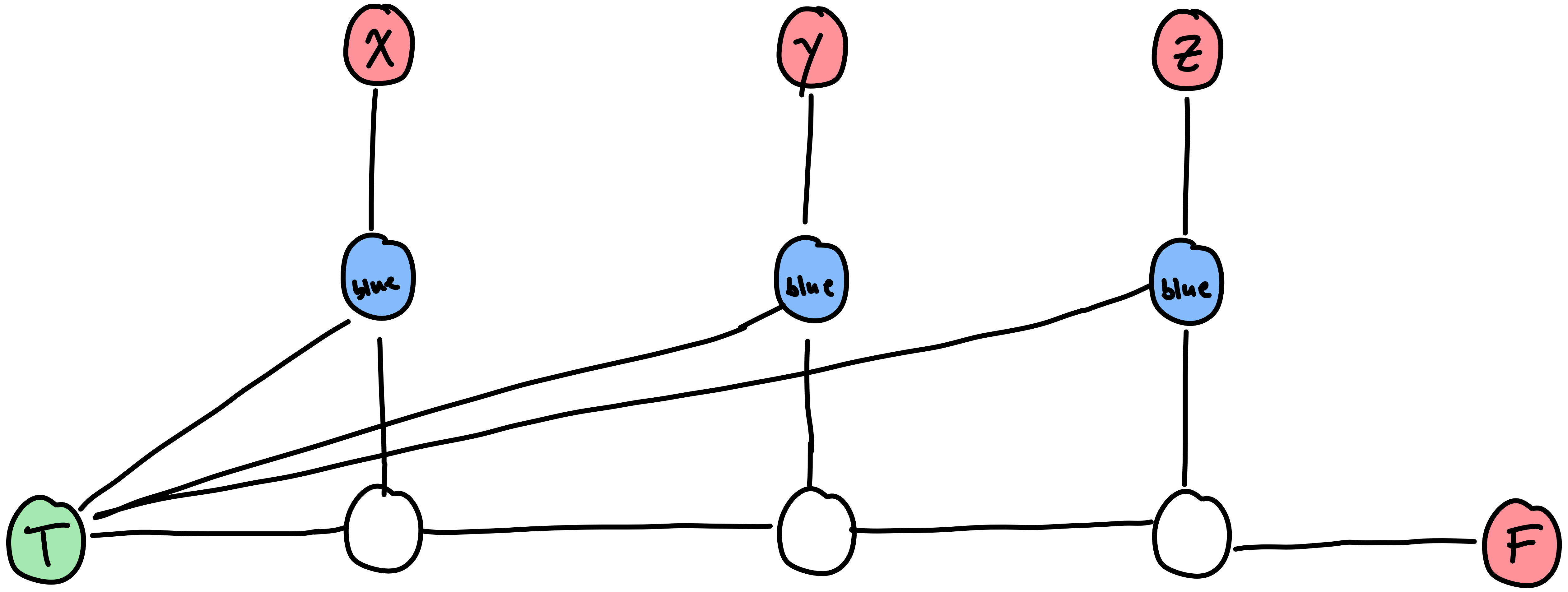


3-color is NP-complete

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

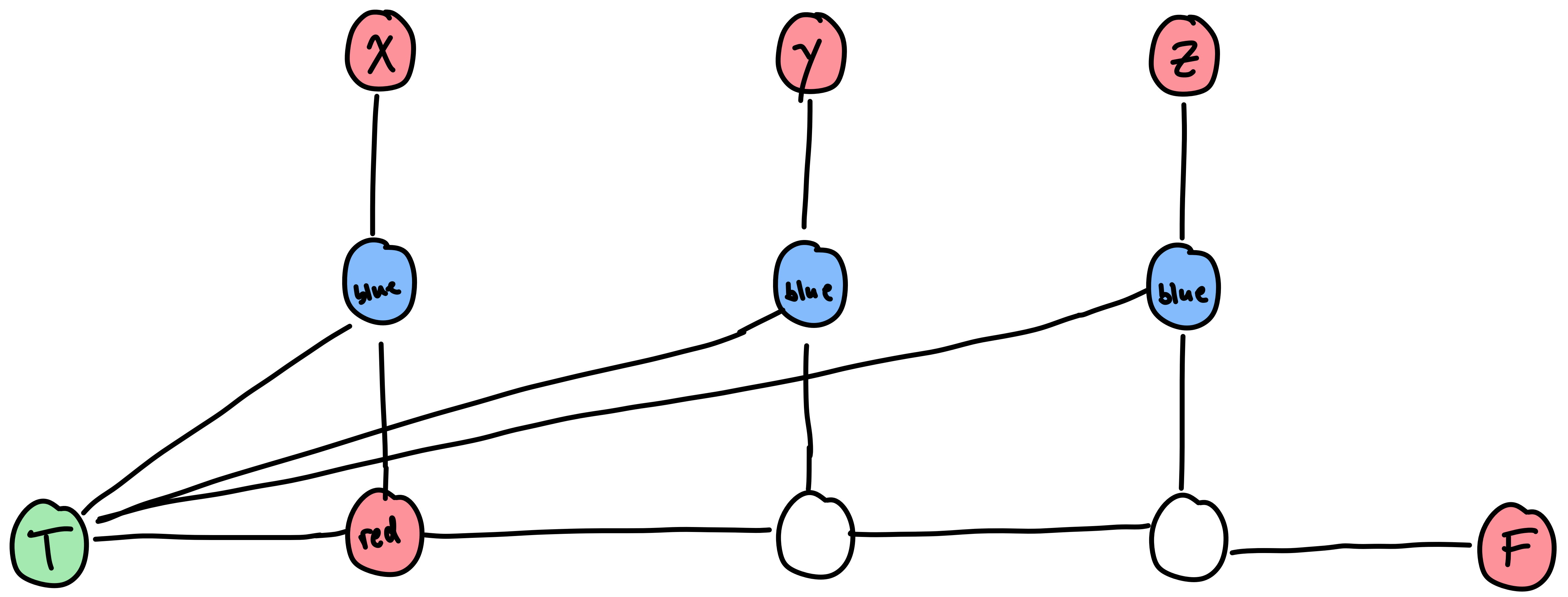


3-color is NP-complete

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

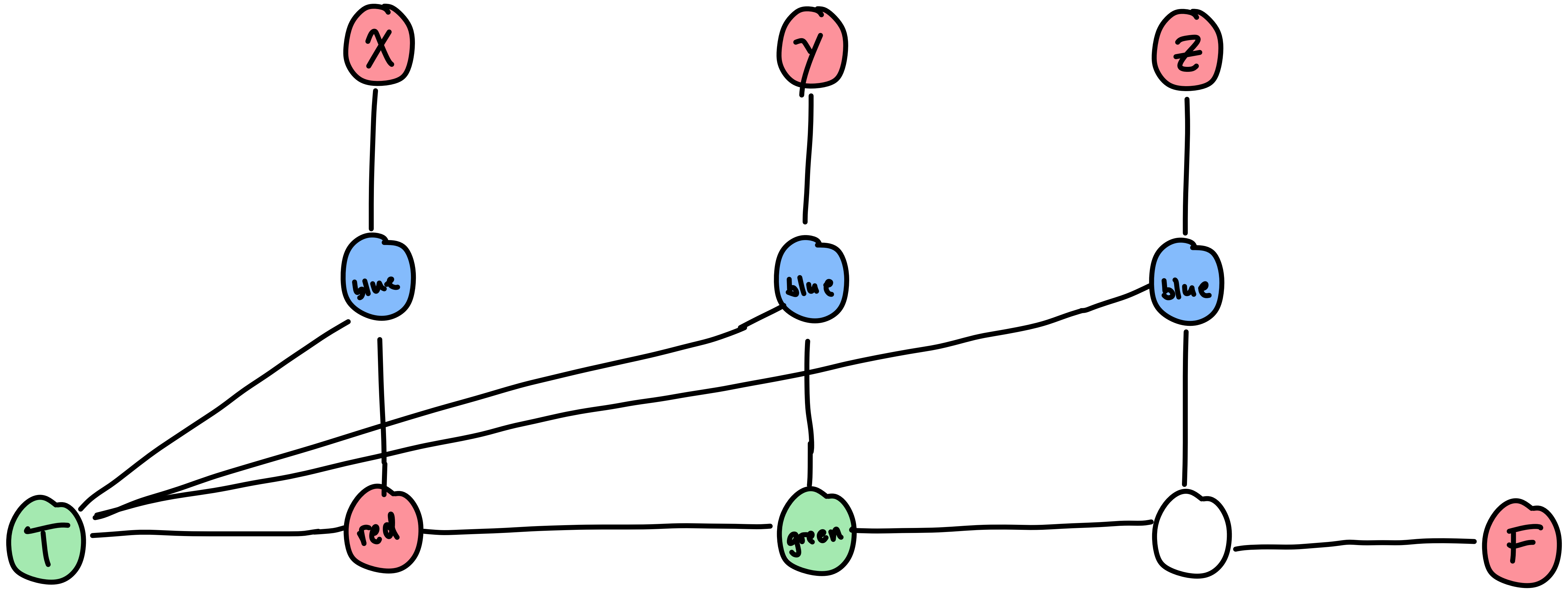


3-color is NP-complete

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

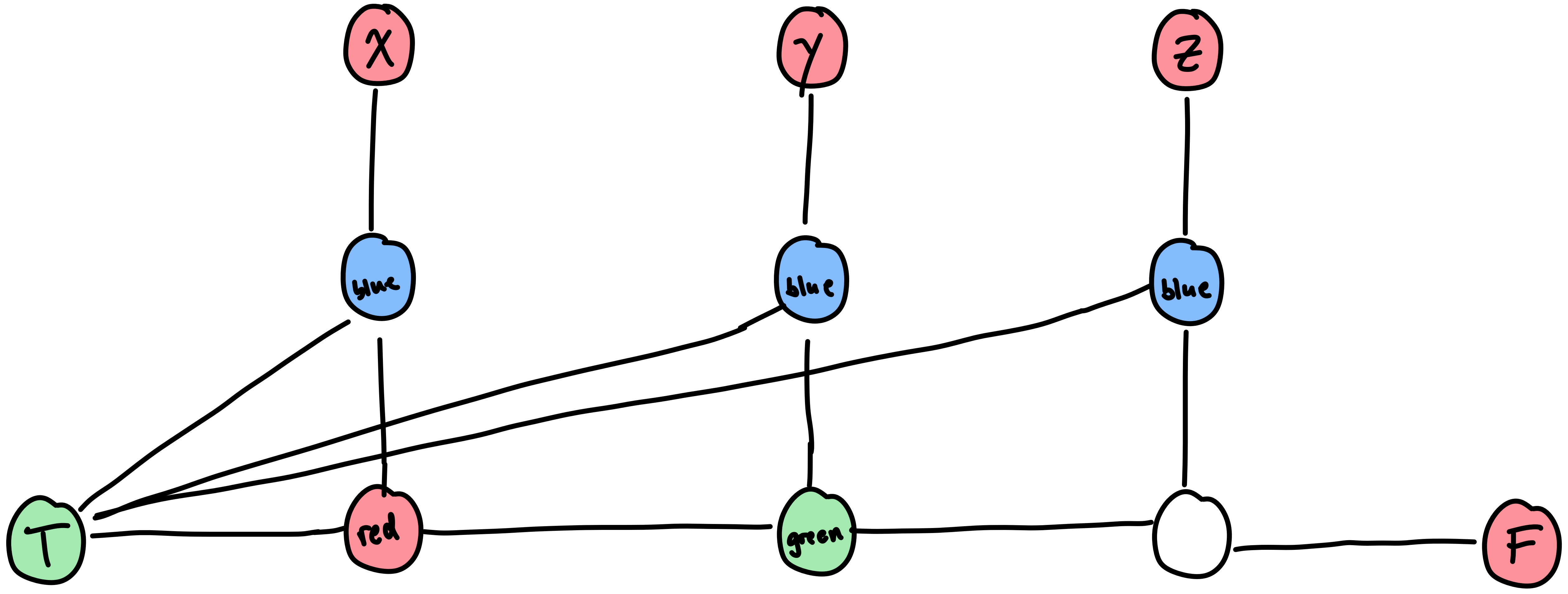


3-color is NP-complete

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



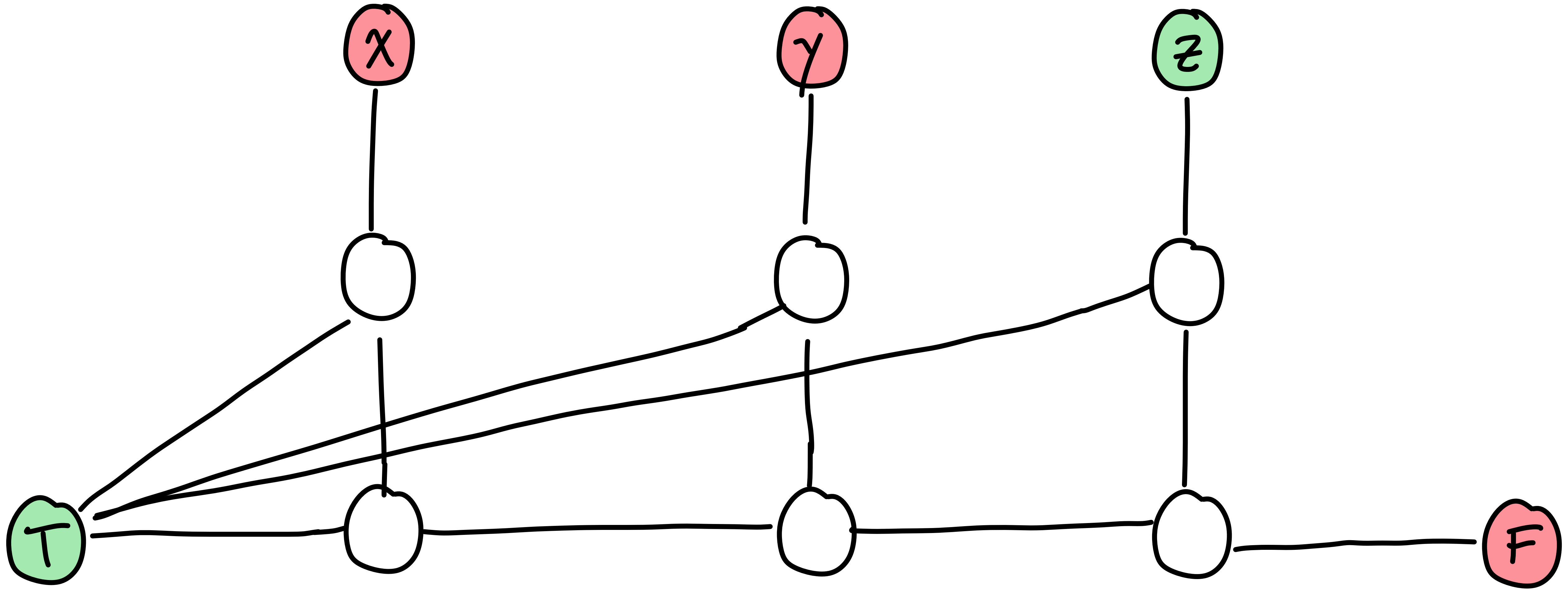
there is no color we can assign!

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

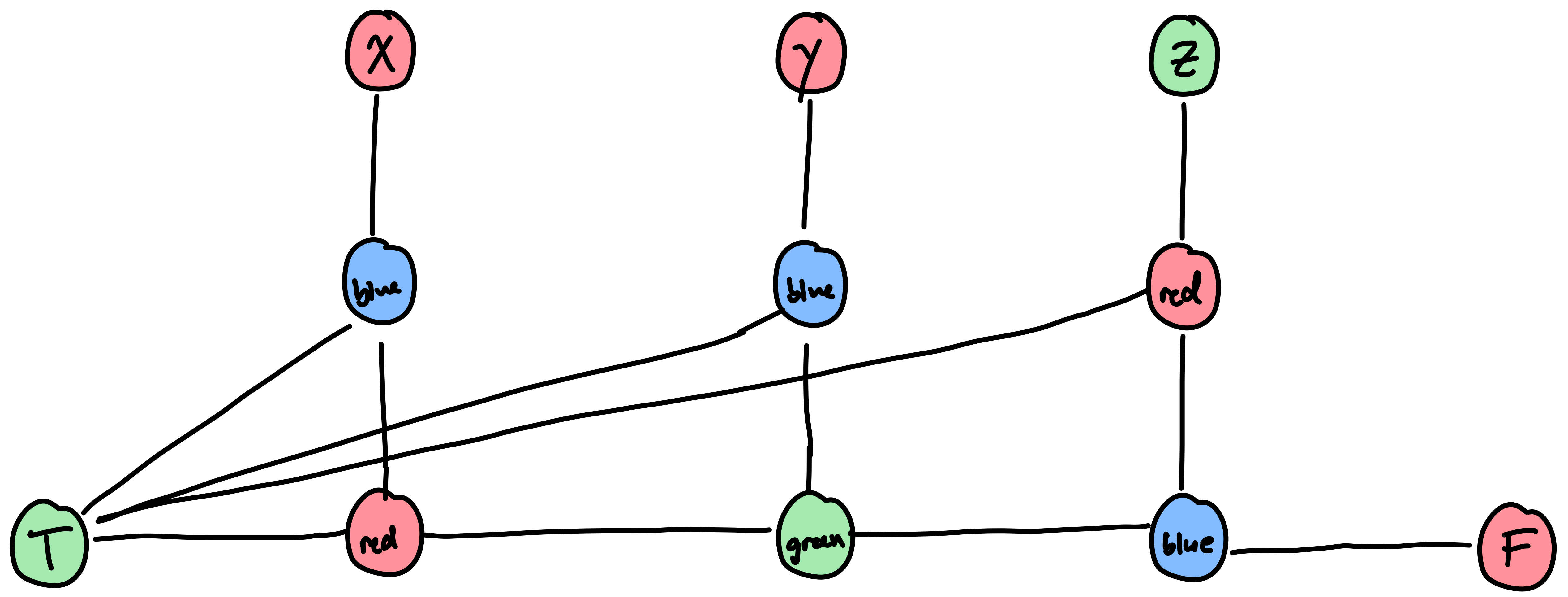


3-color is NP-complete

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

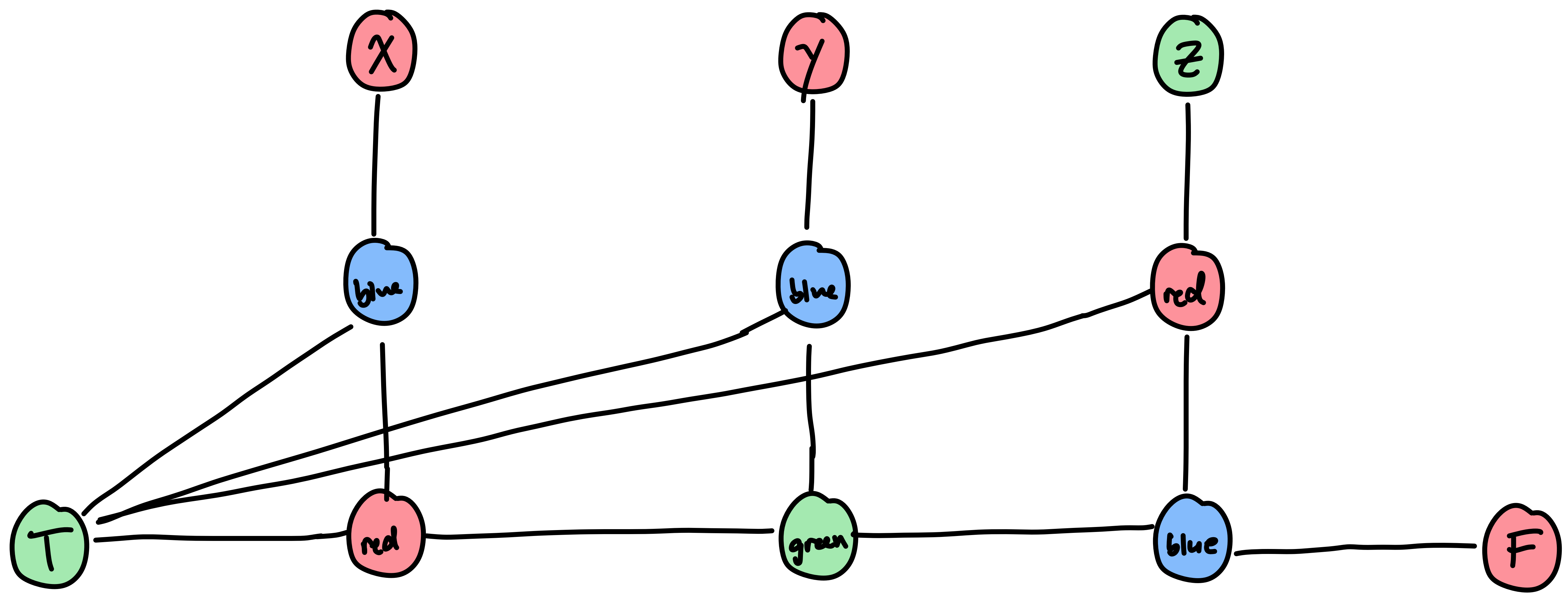


3-color is NP-complete

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

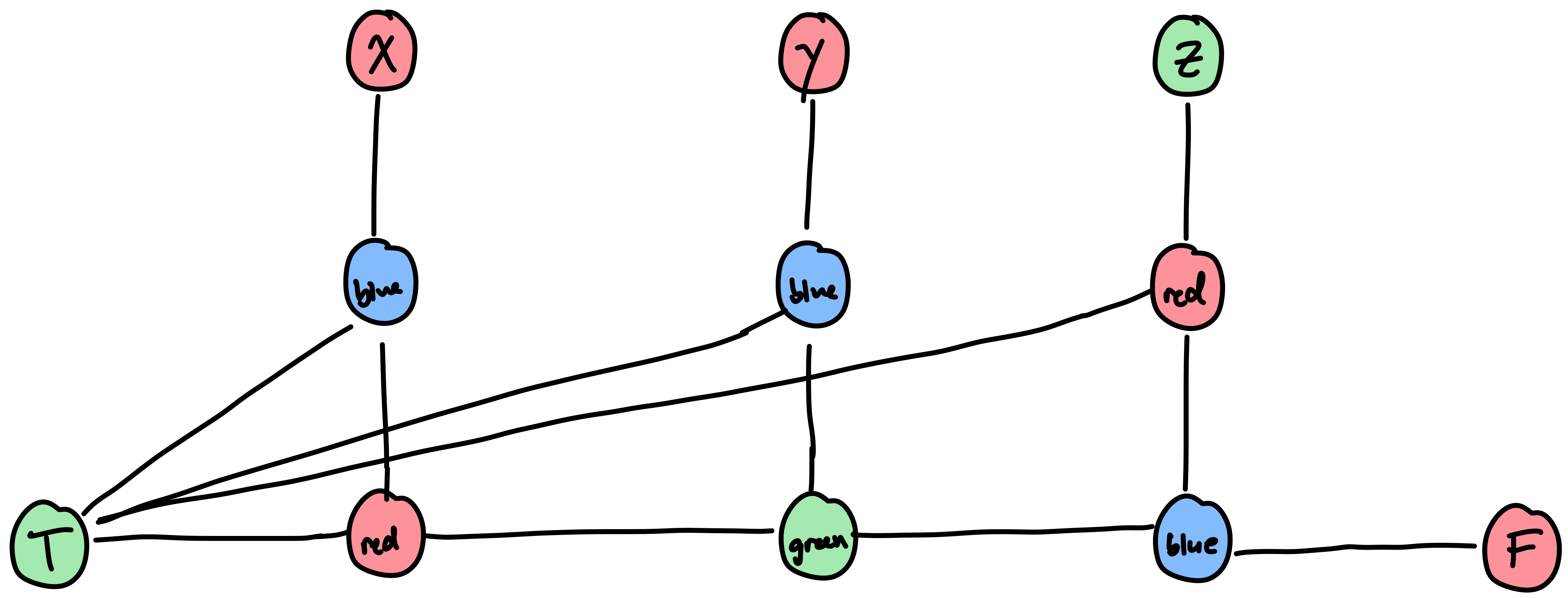


3-color is NP-complete

We now need to construct a "gadget" per clause $x \lor y \lor z$ 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

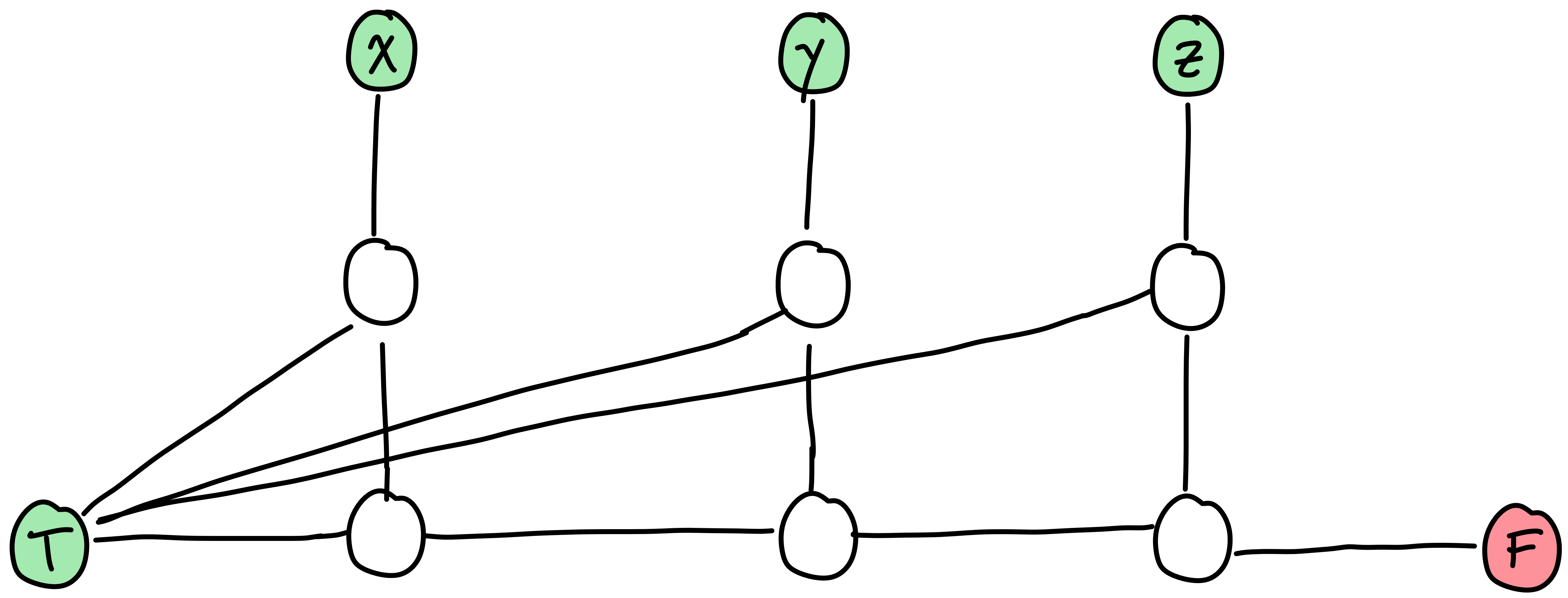


3-color is NP-complete

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

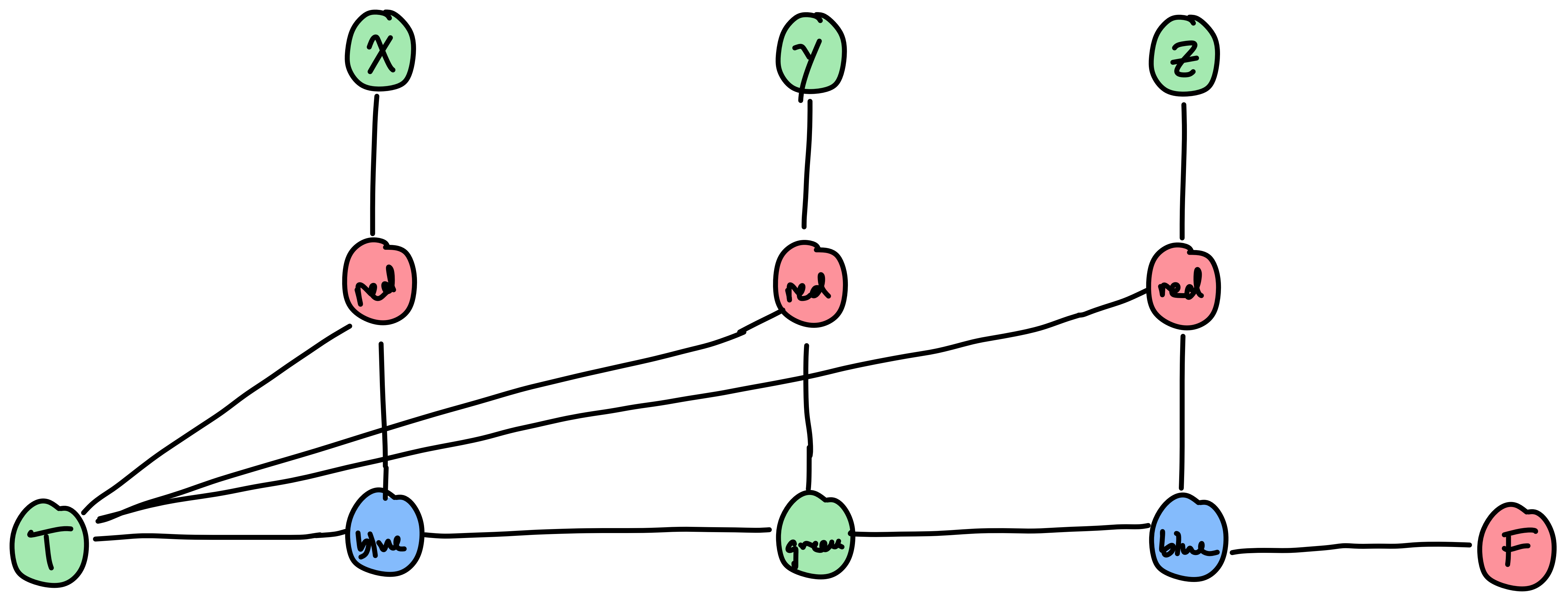


3-color is NP-complete

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 $(B, z_i, \neg z_i)$ 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 $\neg z_i$ 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

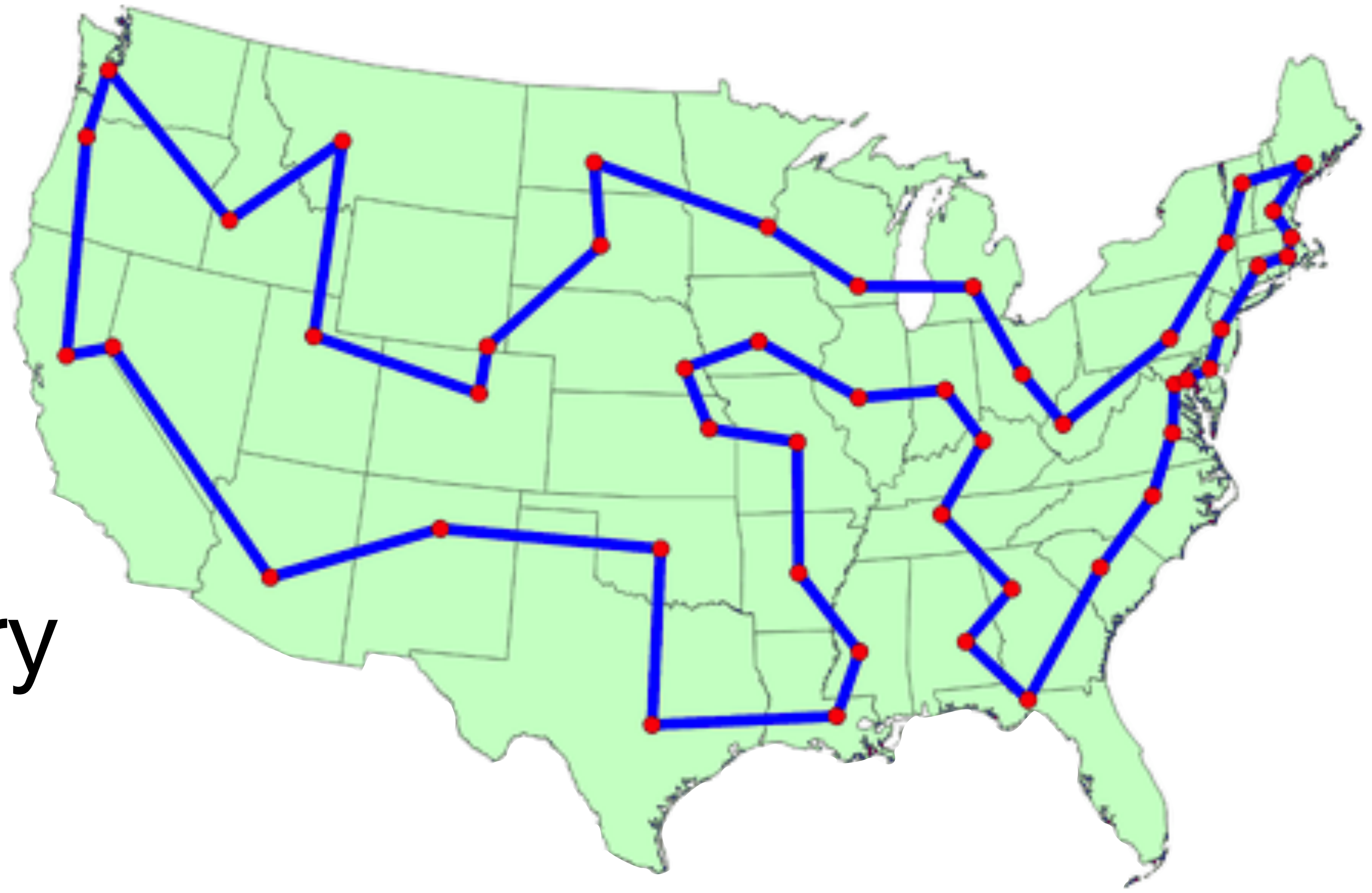
3-color is NP-complete

Putting it all together

- **Reduction proof:**
  - “Yes” → “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” ← “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

Traveling Salesman problem

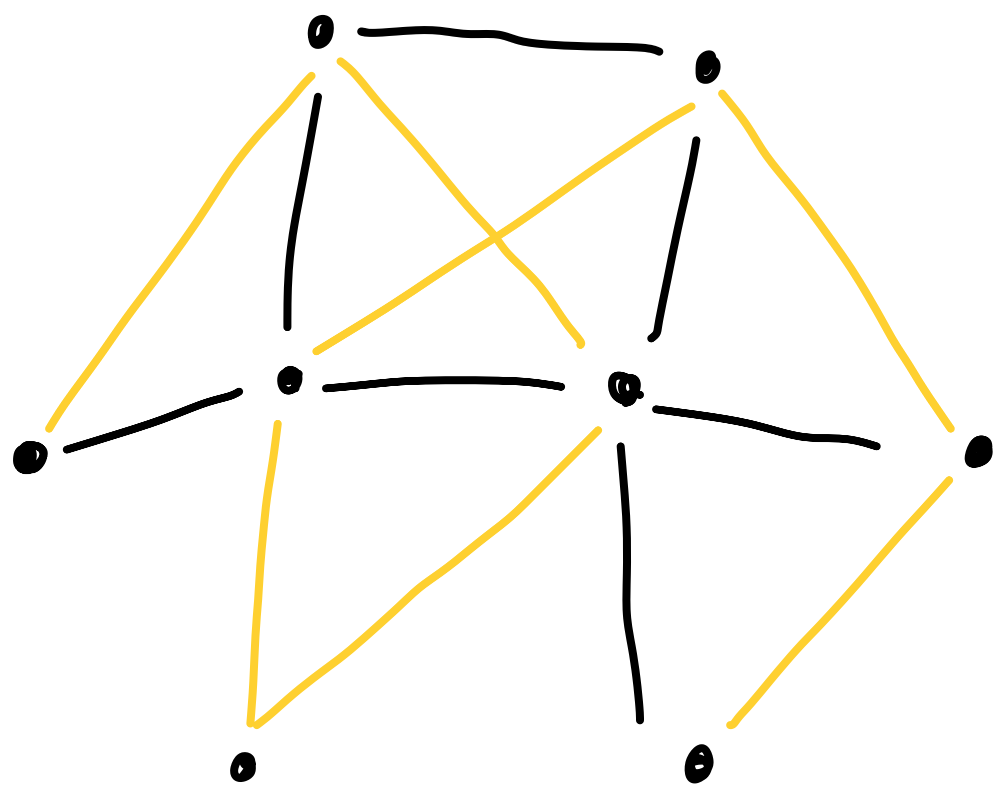
- **Input**: graph $G = (V, E)$, weight function $d : E \to \mathbb{R}^+$, parameter $D \in \mathbb{R}$
- **Output**: If there exists a path visiting all vertices V such that the net distance traveled $\leq D$.
- Traveling Salesman $\in$ NP: Check path π visits every vertex and total length is $\leq D$
- 3-SAT $\leq_p$ Hamiltonian-Path $\leq_p$ Traveling Salesman



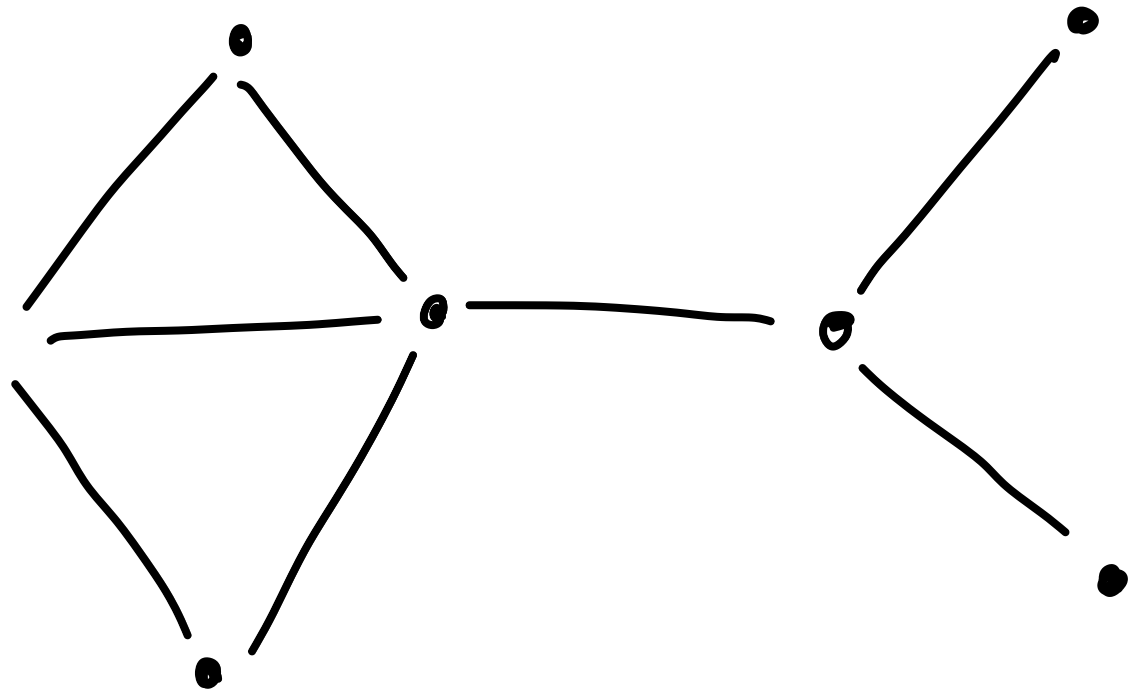
Hamiltonian Path (Directed)

- **Input**: unweighted directed graph $G = (V, E)$
- **Output**: If there exists a path that visits every vertex exactly once
- We saw that it is in NP already
- To prove it is NP-complete, we will need to construct a graph G such that the valid path "encodes" the satisfying assignment to a 3-SAT formula

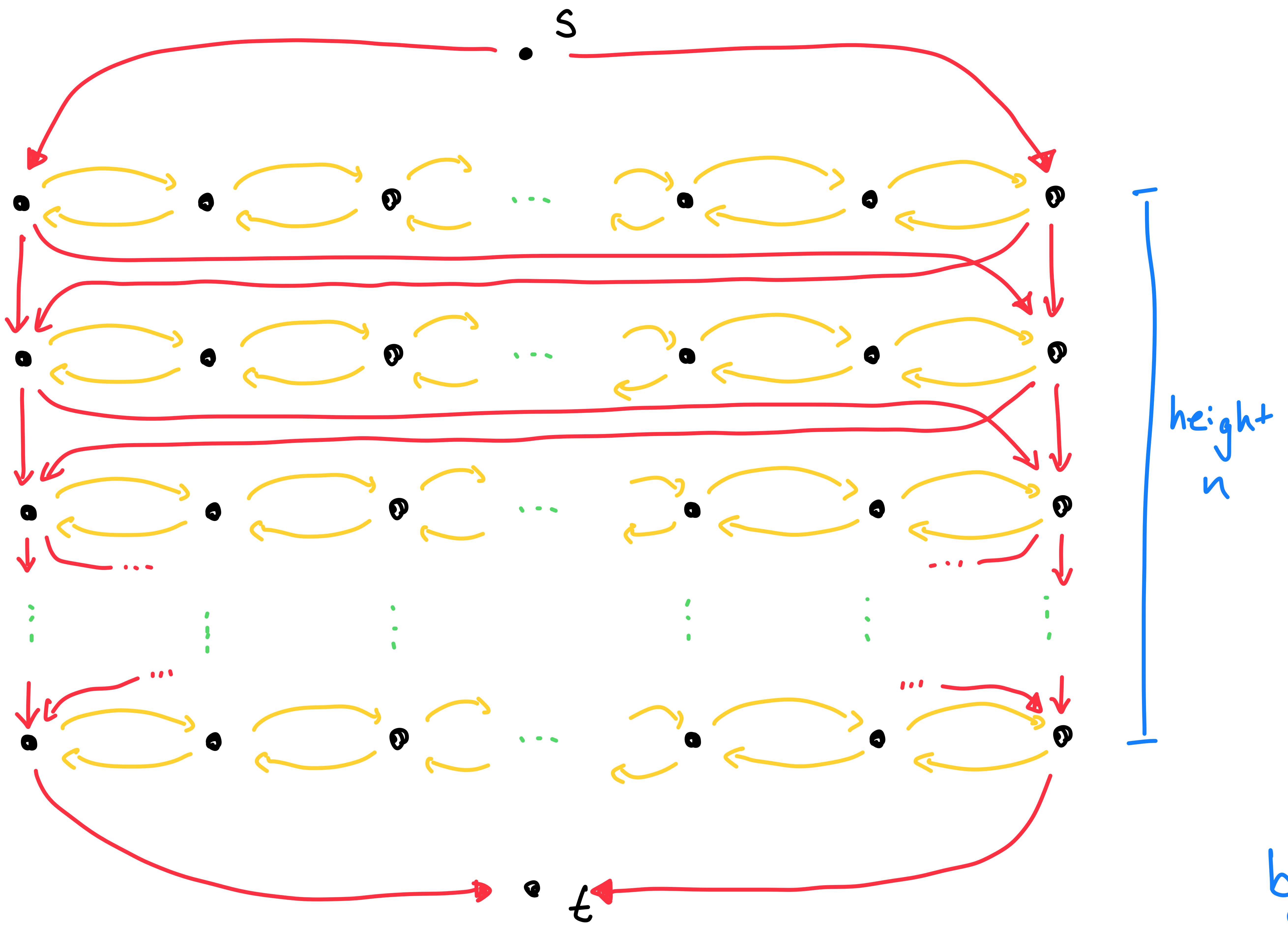
YES



NO



Hamiltonian cycle is NP-complete

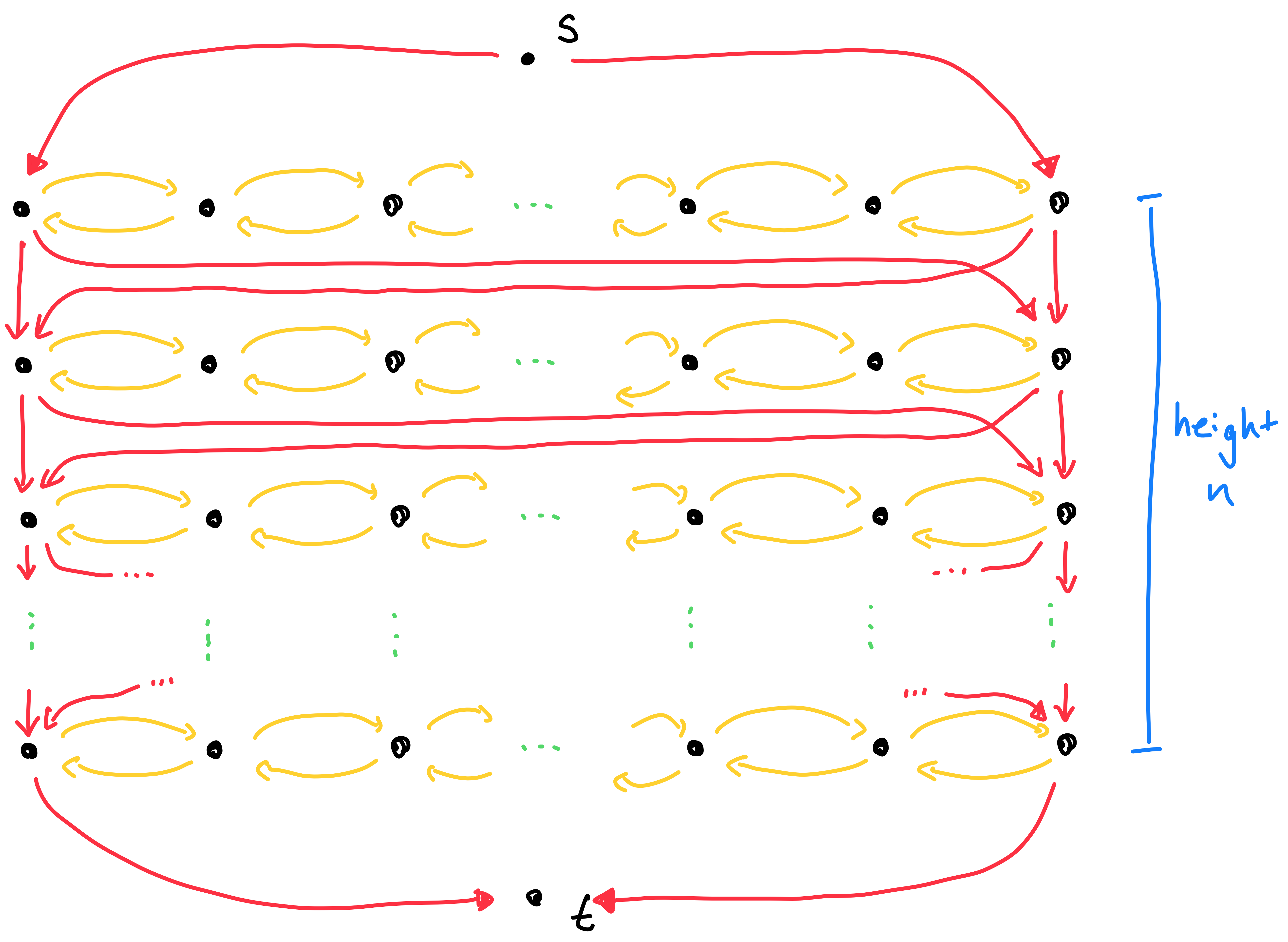


Question: How many different Hamiltonian paths does this graph have?

Answer: 2^n. Each Ham. path is described by the direction the path takes in each row.

bij. between Ham paths and $x \in \{0,1\}^n$.

Hamiltonian cycle is NP-complete



Intuition for reduction:

Add gadgets to graph corresponding to clauses in the 3SAT formula.

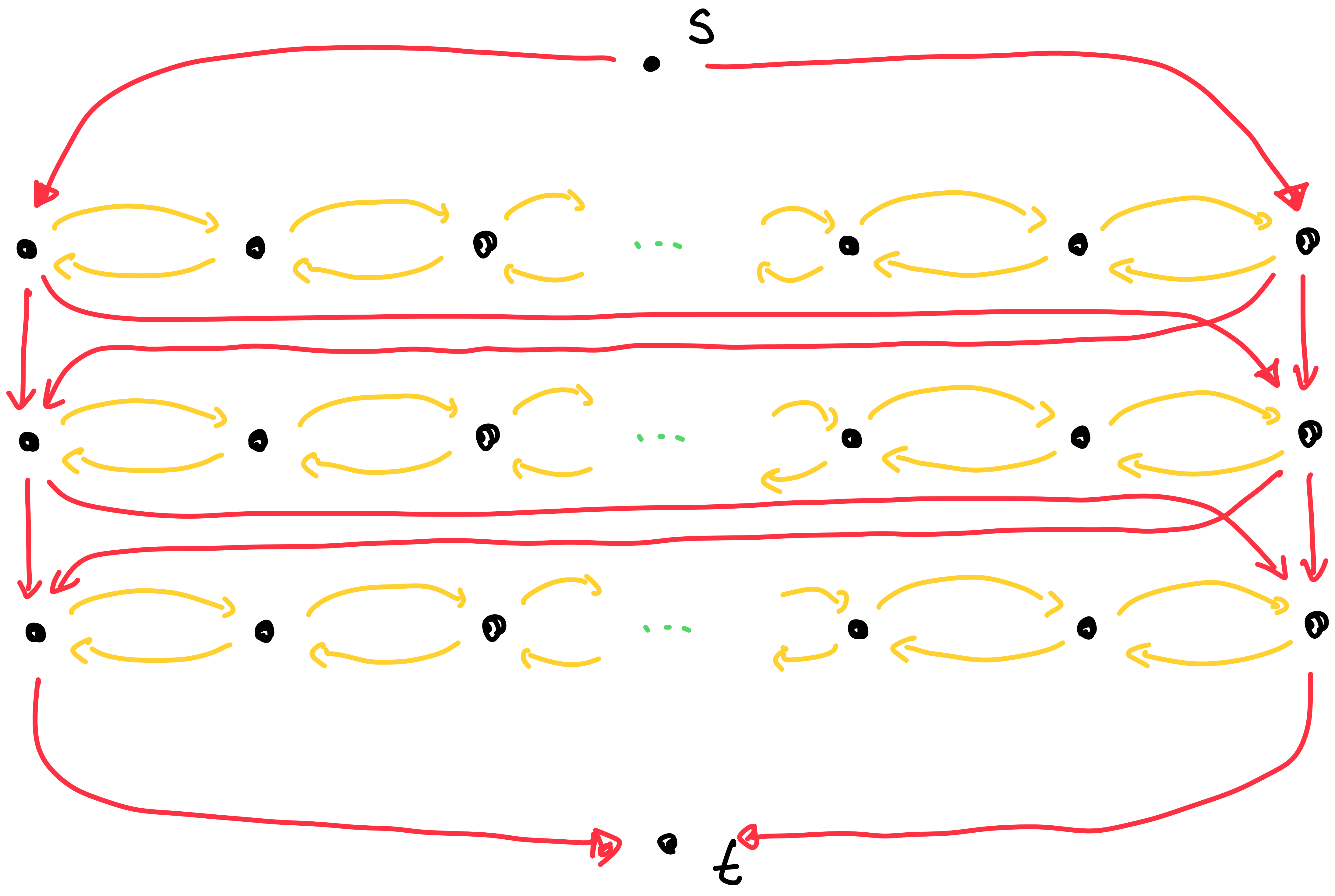
Yes ⟺ Yes intuition:

$x_i = 1$ in 3-SAT formula

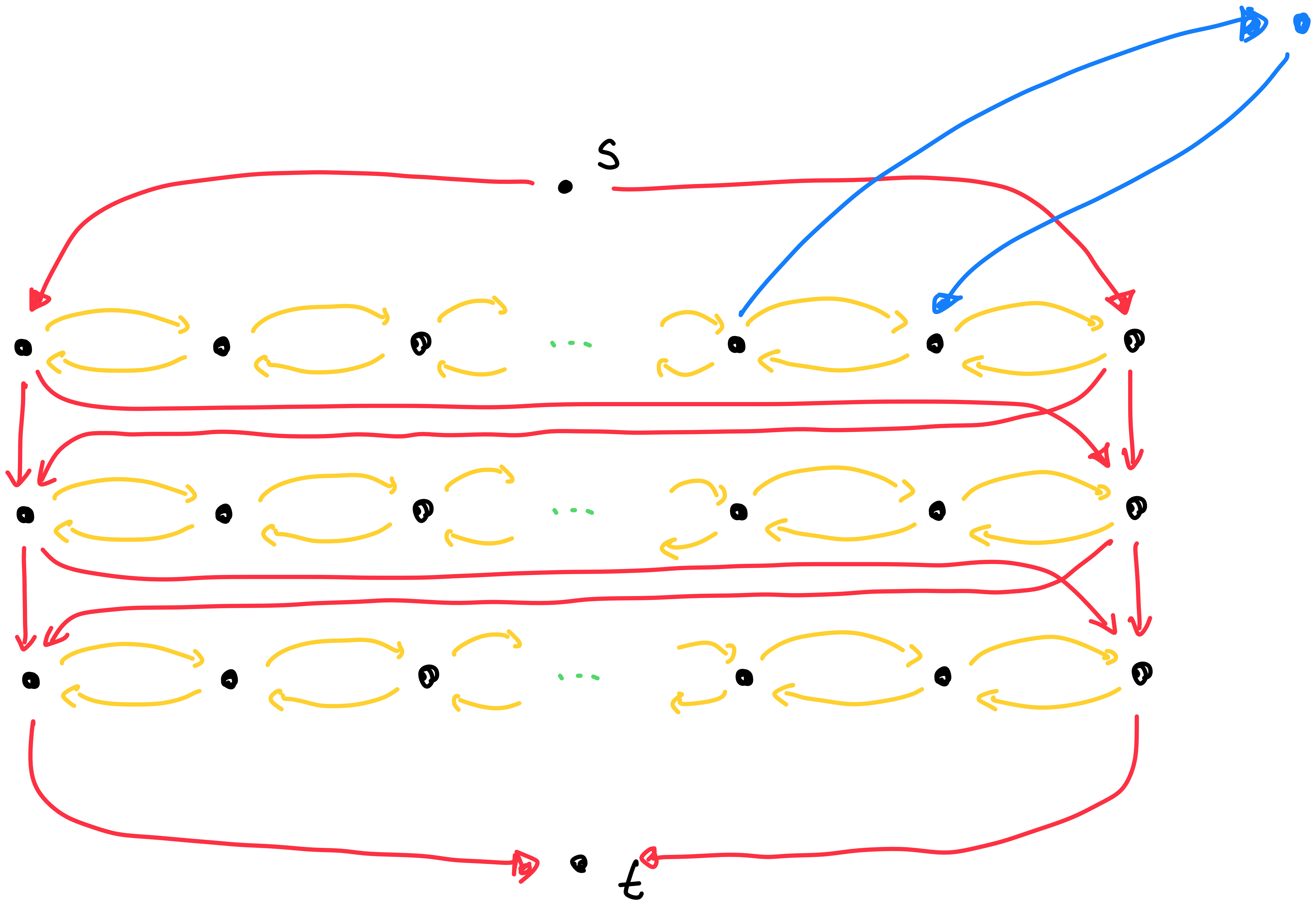
iff

row i of graph is traversed left to right

Hamiltonian cycle is NP-complete



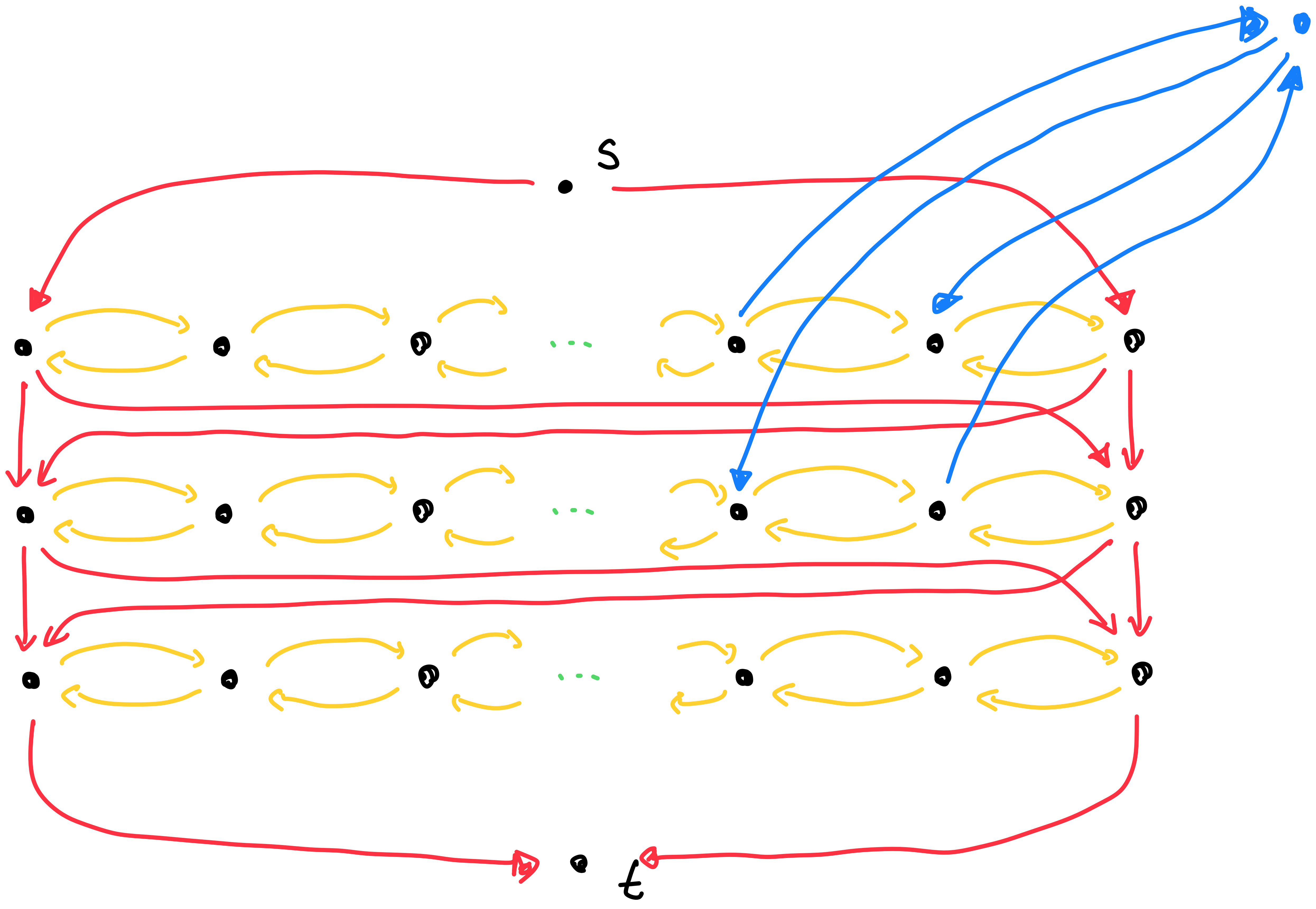
Hamiltonian cycle is NP-complete



If we include this additional gadget then row 1 must go left to right to be a Ham cycle.

$x_1 = 1$

Hamiltonian cycle is NP-complete



If we include this additional gadget then

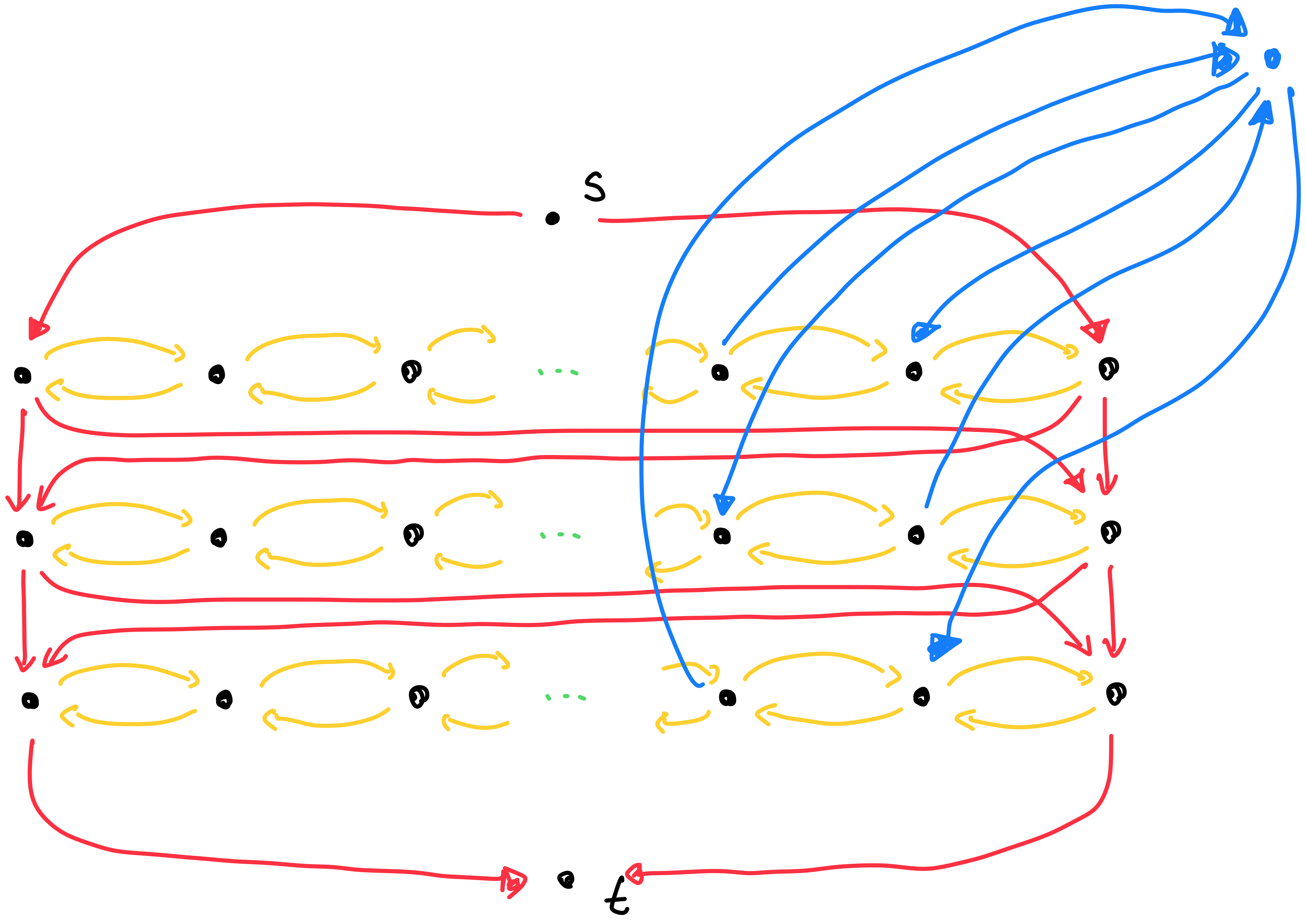
row 1 must go left to right

OR

row 2 must go right to left

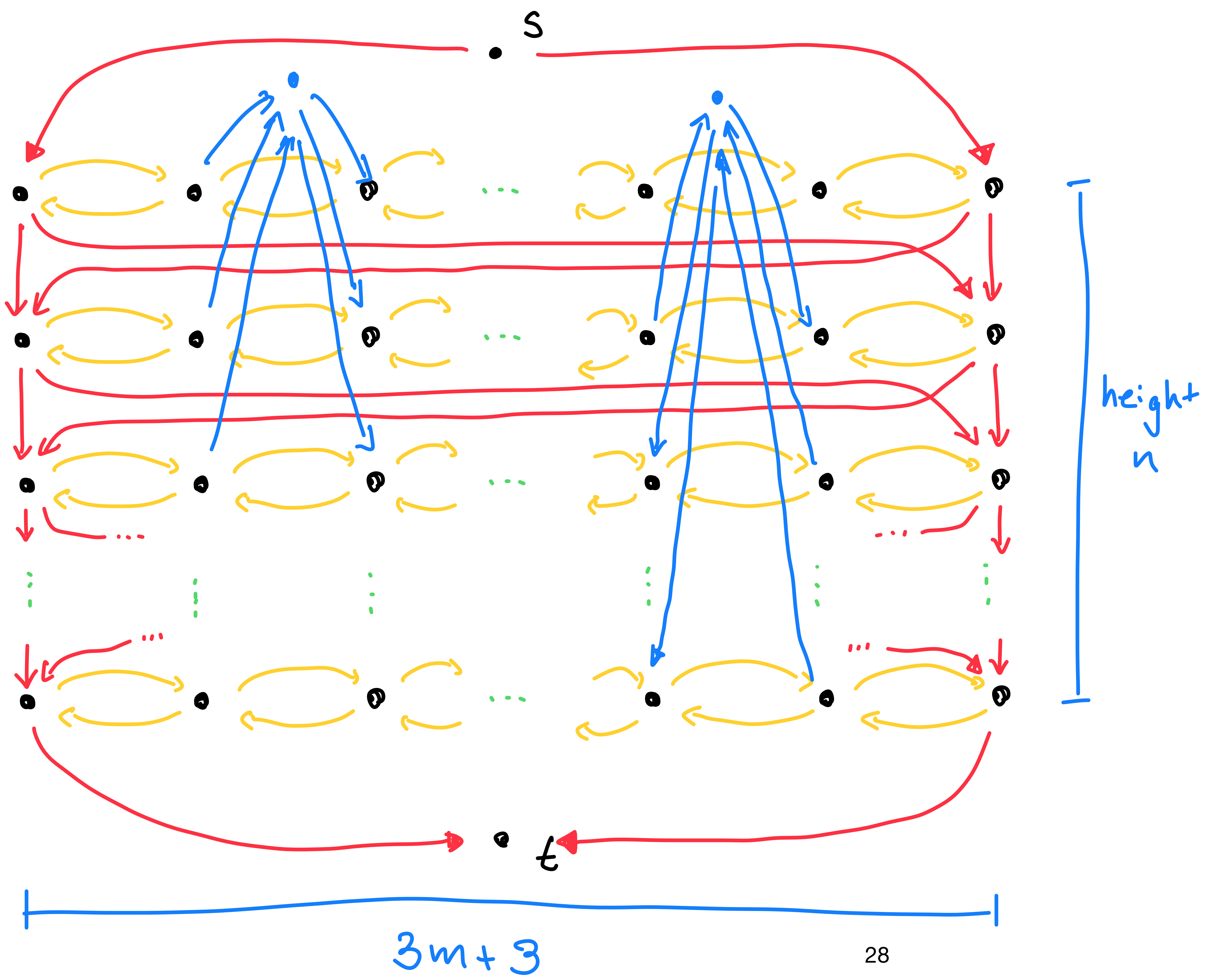
$x_1 \vee \neg x_2 = 1$

Hamiltonian cycle is NP-complete



$x_1 \vee \neg x_2 \vee x_3 = 1$

Hamiltonian cycle is NP-complete



$m = \#$ of clauses

Install gadget for clause j

$\varphi_j = (x_{i_1} \vee x_{i_2} \vee x_{i_3})$

in columns $3j, 3j-1$

and rows i_1, i_2, i_3

so that gadgets never overlap

Hamiltonian cycle is NP-complete

- **Proof of correctness for reduction**:
  - Reduction is easy to see that it is poly-time.
    - Graph has $2 + n(3m + 3) + m$ vertices for formula on n variables and m clauses.
    - Each clause adds 6 edges after standard construction for n variables.
  - Remains to prove that “yes” $\rightarrow$ “yes” and “yes” $\leftarrow$ “yes”.

Hamiltonian cycle is NP-complete

- **“Yes” $\to$ “Yes”:** If φ is satisfiable then a Ham. path exists.
  - Let x be a satisfying assignment for φ
  - For each clause φ_j, identify a literal that is set to true (at least one exists).
  - Construct path $s \rightsquigarrow t$ traversing row i from left to right iff $x_i = 1$ except for a diversion to the vertex φ_j if x_i is the identified literal for clause φ_j.
  - Every variable in every row is necessarily visited and each clause variable must be visited since we identify a true literal.

Hamiltonian cycle is NP-complete

- **“Yes” $\leftarrow$ “Yes”:** If a Ham. path exists φ is satisfiable.
    - Any Ham. path must be from s to t since s is a source and t is a sink.
    - Set $x_i = 1$ iff the leftmost vertex of row i is visited before the rightmost vertex
    - Since each vertex φ_j is visited, some literal in that clause must be set to be true as φ_j is only visited iff the direction of the path is equiv. to a literal in the clause being set to true
    - Therefore, all clauses φ_j are satisfied by the assignment given by x

Decision problems = Optimization problems

- Optimization problem: Find the *shortest path* for the traveling salesman to visit all the cities.
- Decision problem: Decide if there *exists* a path of length $\leq D$ for the traveling salesman to visit all the cities.
- **Theorem:** There exists an efficient algorithm for optimization iff there exists an efficient algorithm for decision.
- **Proof:**
  - ($\Longrightarrow$): Let $\mathscr{A}_{\text{opt}}$ solve optimization. To solve decision, run $\mathscr{A}_{\text{opt}}$ and calculate D^*. Answer if $D^* \leq D$.
  - ($\Longleftarrow$):
    - Let $\mathscr{A}_{\text{dec}}(D)$ solve decision for parameter D.
    - $D_{\max} := \sum_e d(e)$. Starting from $D \leftarrow D_{\max}/2$ use $\mathscr{A}_{\text{dec}}(D)$ to "binary search" to calculate D^*.
    - If $D^* \leq D$, then $\mathscr{A}_{\text{dec}}(D)$ outputs true and if $D > D^*$, then $\mathscr{A}_{\text{dec}}(D)$ outputs false.
    - Tota runtime is $O\left(\log(D_{\max})T_{\mathscr{A}_{\text{dec}}}\right)$ which is polynomial in input length.

The value of 3SAT

- By now it should have become pretty clear the value of 3SAT
- It has just enough versatility that we can prove that it is NP-complete
- It has just enough structure that we can use it to prove NP-completeness for a wide range of problems
- Really a beautiful problem!

P vs NP recap

- **NP** is the class of decisions problems for which for which every "yes" instance has a certificate (or witness) that can be verified in polynomial time.
- **Theorem:** If a problem X is NP-complete, then the problem has a polynomial-time algorithm iff P $=$ NP.
  - I.e., we have proven the equivalence of the difficulty of all NP-complete problems.
- **General procedure for proving Y is NP-complete:**
  - Prove Y is in NP
  - Find an X that is NP-complete and then show $X \leq_p Y$.
- $X \leq_p Y$ if there is a poly-time map f s.t. x is a "yes" instance of X iff $f(x)$ is a "yes" instance of Y.
  - Proving a reduction involves construction f and arguing the following two statements:
  - x is a "yes" instance of $X \implies f(x)$ is a "yes" instance of Y.
  - $f(x)$ is a "yes" instance of $Y \implies x$ is a "yes" instance of X.