3Coloring, Mario.

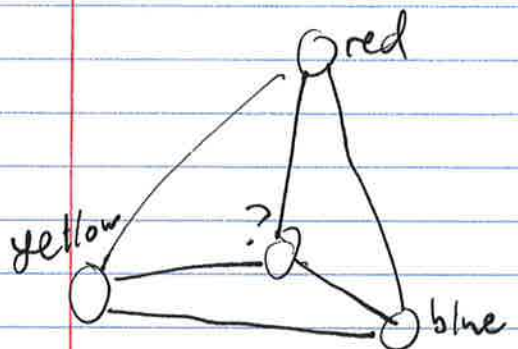
We did an intro lecture on complexity, we did a lecture on boolean NP-complete problems with boolean formulas. We did a lecture on NP-complete problems with graphs. Then we did a lecture on NP-complete problems with constraint-like problems like subset-sum and Knapsack. This lecture we will do puzzles and (single-player) games.

You may have some intuition as how the best algorithm you can think of for SAT is really checking all assignments in 2^n time. This may feel like sudoku, for some hard puzzles. It may seem that most similar puzzles are NP-complete. Note, most two-player games are not NP-complete.

First we cover vertex-cover here, see notes from the graphs lecture.

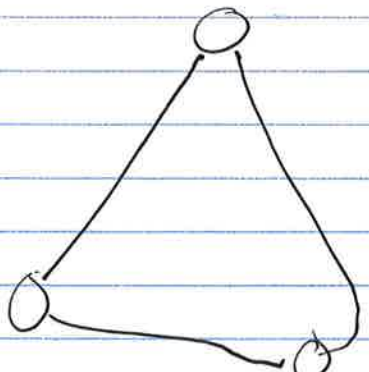
$$3COL = \{\langle G \rangle \mid G \text{ a graph admits a 3-coloring}\}$$

a graph is 3-colorable if you can color the vertices using three colors such that no adjacent vertices share the same color.



K_4

no 3 coloring



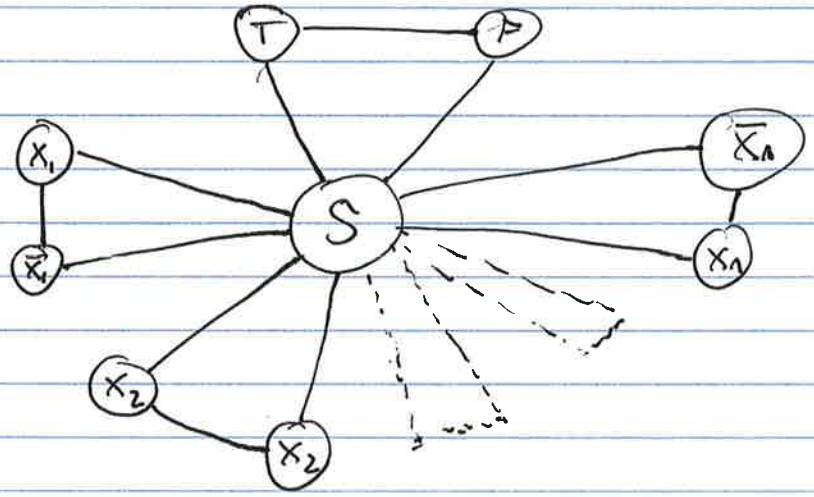
K_3

has a 3 coloring

3COL is NP-complete. While it may be tempting to try to reduce from a graph problem such as Clique, we reduce from 3SAT. For any 3CNF formula, we shall construct a graph which admits a coloring only if the formula was satisfiable.

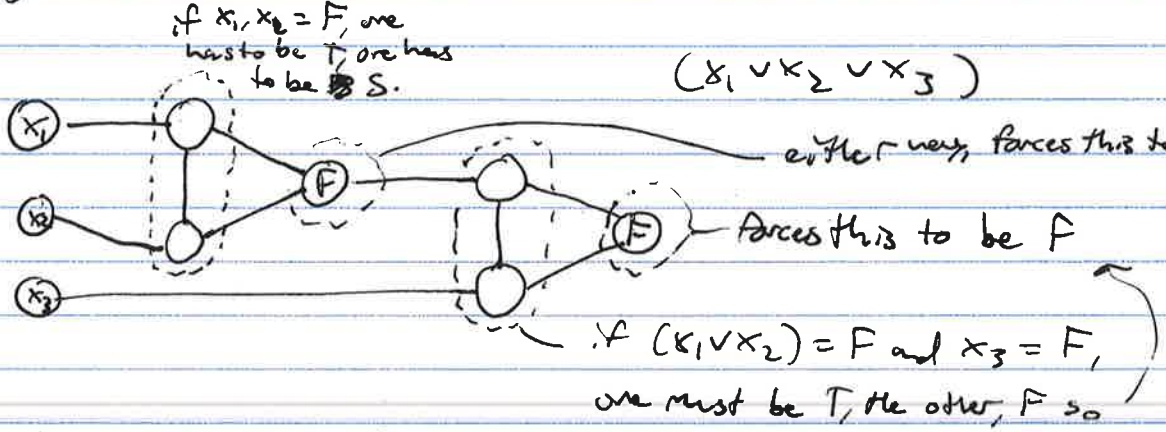
First we show 3COL $\in$ NP. Our verifier on input problem = $\langle G \rangle$ and witness = coloring of the variables checks in polytime if there are 3 colors and if the coloring has the property that no two adjacent vertices have the same color.

Now we transform ϕ into a graph. We want our graph to imitate the following structure. One of $x_i, \overline{x_i}$ is on $\forall x_i \in x_1 \ldots x_n$. And each clause C_i is satisfied. We use three colors, Tangerine for true, ... Fuschia for false, and I dunno. Sapphire for "secret third color". We will use color S only to control colors T and F. To have the first property, that exactly one of $x_i, \overline{x_i}$ is on for each x_i, we build the following graph.

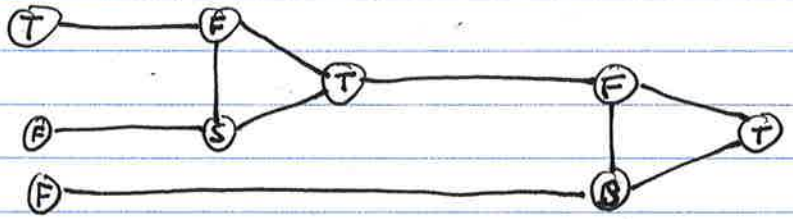


convince yourself $x_i, \overline{x_i}$ cannot both be color T or F. None can be color S.

This is not the entire graph, but it does force an assignment to exist. Now we force the rest of our graph such that every single clause is satisfied. We will build an OR gadget so that we can OR the colors of the clauses together. Later we check if the AND of these clauses is the color T

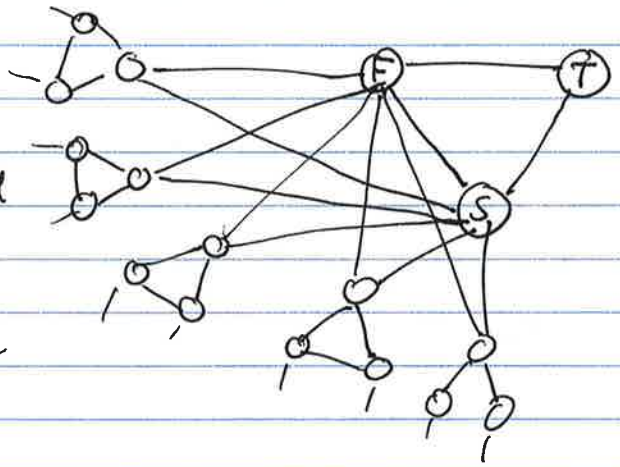


so if $x_1 = x_2 = x_3 = F$, this graph has no coloring of the output node except F



if one of x_1, x_2, x_3 is true, then there exists a coloring with the output node to be true. Doesn't matter that many more colorings exist which don't have the output node as true.

remember the og triangle? we map the output node of each OR gadget to the F and S colored ones of the original triangle. This forces any 3coloring of the graph to have all output nodes the same color, essentially ANDing them together.



We now prove our reduction. It may have been obvious since we had to explain it.

If $\phi \in 3SAT$, $\exists$ a satisfying assignment of its variables. Color the corresponding $x_i, \bar{x}_i$ vertices accordingly and the rest of the colors essentially collapse, forcing the coloring of the graph. We see $G \in 3COL$, since there exists such a coloring.

If $\phi \notin 3SAT$, $\exists$ atleast one unsatisfied clause. Notice for the OR gadget corresponding to this clause, every possible coloring has the output node the same color as its input. The structure means if any other OR gadget has true color, the graph is ~~unsatisfiable~~ not 3colorable. ~~If all clauses~~ If all clauses were false, ϕ was never not in 3SAT, it would have been satisfiable. So $G \notin 3COL$.

$f(\phi) = G$ is computable in poly time, few for loops over ϕ.

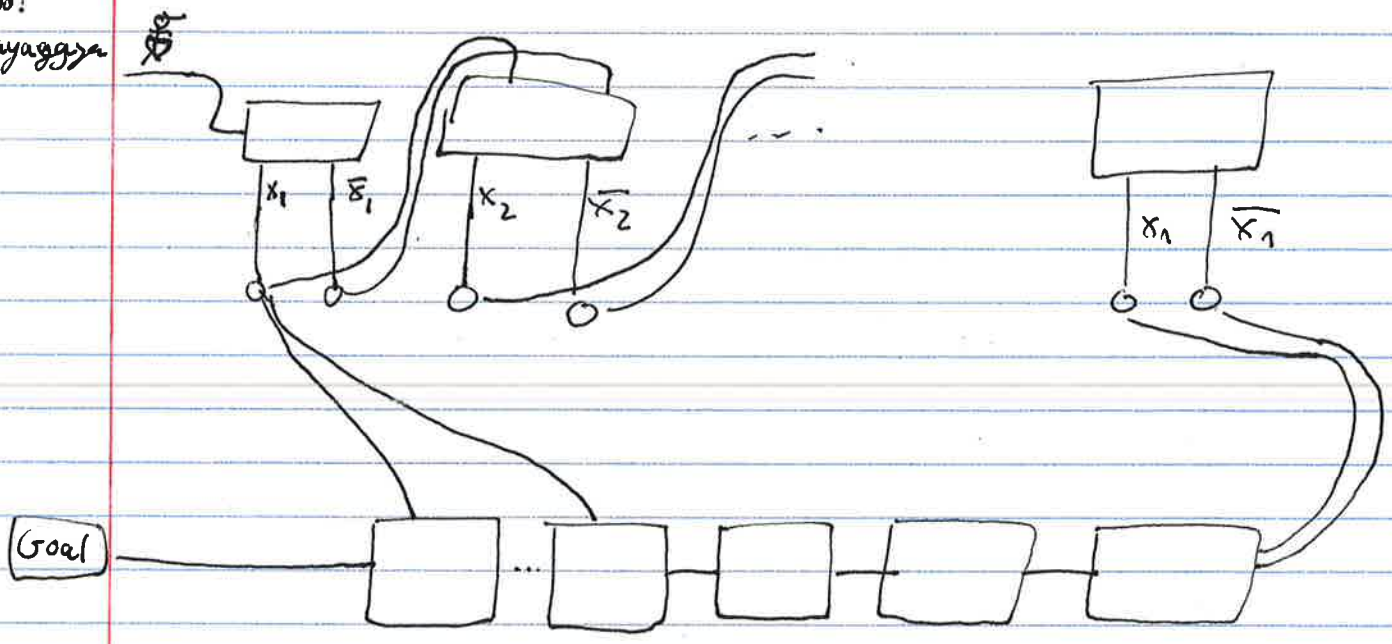
We see $3SAT \leq_p 3COL$. Since 3SAT is NP-complete and we proved 3COL is in NP, we conclude 3COL must also be NP-complete.

List of NP-complete (:) games

- Mario (probably not $\in$ NP)
- Battleship
- Sudoku on an $n^2 \times n^2$ grids w $n \times n$ blocks
- wordle (NP-hard, prob not in NP)
- Minesweeper
- Mastermind
- Freecell Solitaire
- Tetris
- [redacted]
- [redacted]

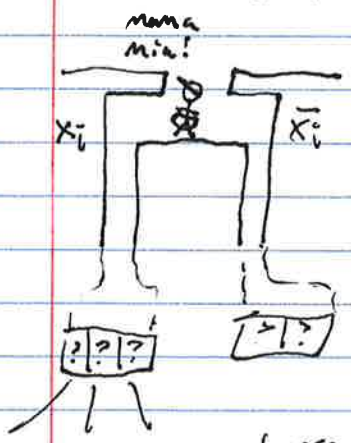
lets prove Mario is NP-complete. This comes from a great paper, a serious one, by Erik Demaine I have recently been told of a good video explanation of this reduction as well. search "Mario SAT NP-hard" or something on youtube. We prove $3SAT \leq_p MARIO$. We construct our level like so

yahoo!
yayayaggya

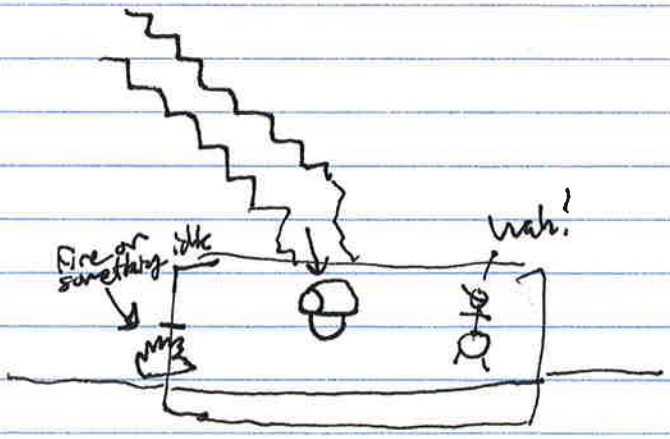


basically each variable gadget, he can only choose one of x_i or $\bar{x}_i$. He will send a mushroom into each clause gadget. he can only go from one clause to the next if he had a mushroom.

variable gadget



he can only choose one, but he sends mushrooms.



needs atleast one mushroom to get through the fire door.

If $\phi \in 3SAT$, then the level is satisfiable, Mario takes the chutes according to the exact assignment of the variables. so $L \in MARIO$.

If $\phi \notin 3SAT$, some clause is always unsatisfied. Mario will get stuck in this clause gadget, burn and die with no mushroom so we see $L \notin MARIO$.

Thus $\phi \notin 3SAT \iff L \notin MARIO$. This transformation takes polytime so we observe $3SAT \leq_p MARIO$ and that Mario is NP-hard.

Why is this formalization of mario not NP-complete? Why is Mario not in NP? Turns out this formalization is only NP-hard and not known to be NP-complete. while it is sedimthe to consider a witness for a verifier as a sequence of button pushes, this verifier runs in polytime only if the witness is in ~~NP~~ polynomially sized, which we don't know is true.