CS 3510 Algorithms 11/01/2023

Lecture 18: 3-Coloring and Mario

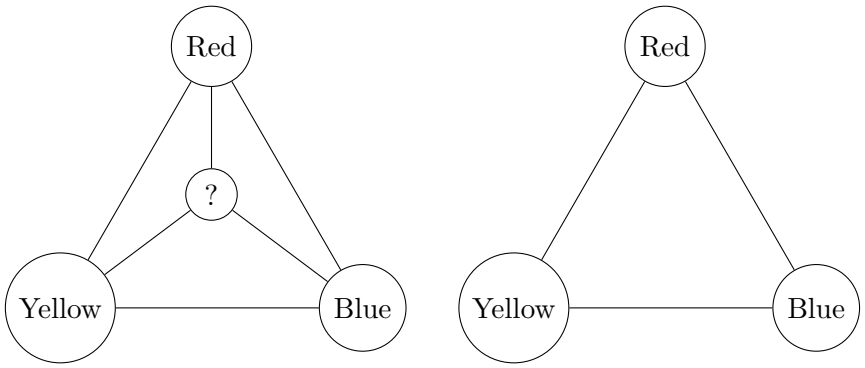
Lecturer: Abrahim Ladha *Scribe(s): Jiaxuan Chen*

3-Coloring

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

You may have some intuition on how the best algorithms you can think of for SAT are really checking all assignments in 2^n time. This may feel like sudoku, for some hard puzzles. If you've seen that most similar puzzles are NP-complete. Note, that most two-player games are not NP-complete.

$3COL = \{\langle G \rangle \mid G \text{ admits a 3 coloring}\}$
A graph is 3-colorable if you can color the vertices using three colors such that no adjacent vertices share the same color.

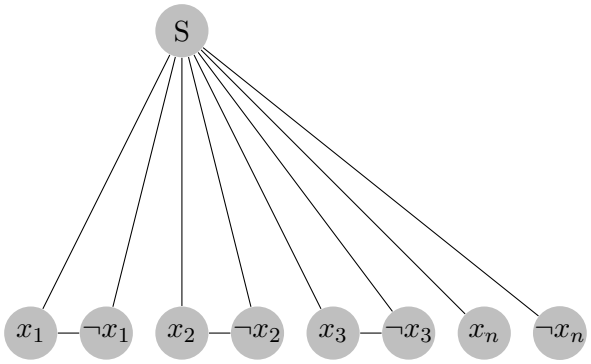


K_4 is not 3-colorable. K_3 is 3-colorable.

$3COL$ 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 that admits a 3-coloring if and only if the formula was satisfiable.

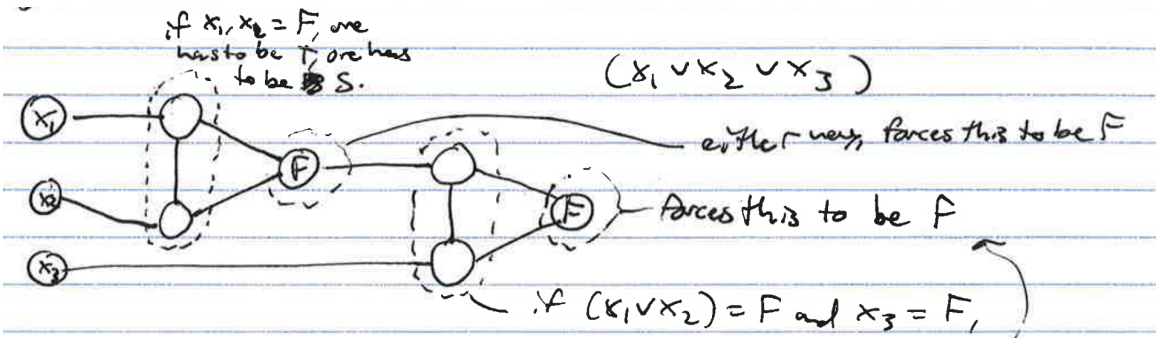
First we show $3COL \in \mathsf{NP}$. Our verifier on input problem G and witness a coloring of the variables checks in polytime if there are 3 colors and if the coloring has the property that no two adjacent vertices have the same color.

Now we transform Φ into a graph, we want our graph to replicate the $3CNF$ structure, one vertex for each x_i and $\neg x_i$, $i = 1 \dots n$. And each clause C_i is satisfied. We use three colors; tangerine for true, fuchsia for false, and I choose sapphire for "secret third color". We will use color S only to control colors T and F. To have the first property, that each of x_i, $\neg x_i$ is T for each x_i, we build the following graph.

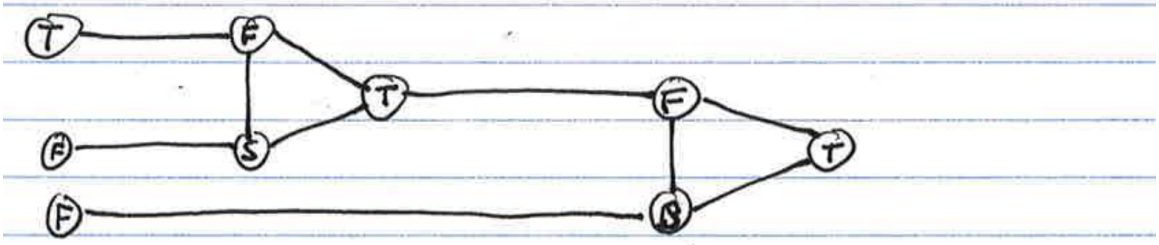


Convince yourself x_i, $\neg x_i$ cannot both be color T or both F. None can be color S. This simulates that every variable has an assignment, and that $x_i = 1 \iff \neg x_i = 0$

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 tie the colors of the clause together. Later we check if the AND clauses fit the color T.

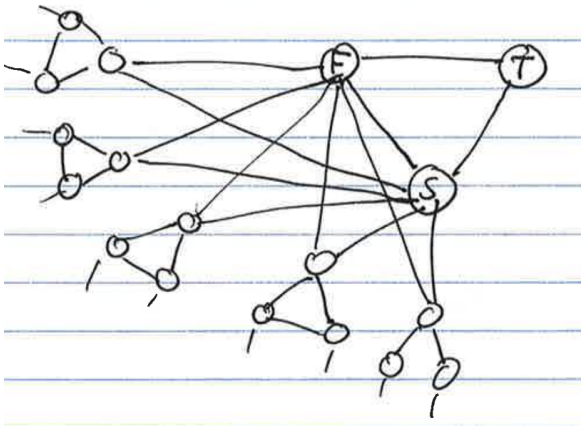


so if $x_i = x_k = x_j = F$ this graph has no coloring of the output node except F.



If one of x_i, x_k, x_j is true, then there exists a coloring with the output node to be true. Doesn't matter that there are colors that 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 my 3coloring of the graph to have all output nodes for some color, essentially AND'ing them together.



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

If Φ is 3SAT, there is a satisfying assignment of n variables. Color the corresponding x_i vertices accordingly and the rest of the clause essentially collapse. Any false coloring will not propagate, hence $G \in 3COL$ since there exists such coloring.

If Φ is not 3SAT, there is at least one unsatisfied clause. No true for the OR gadget corresponds to this clause, every possible output node of that gadget must be the false color. The structure means no other can be gadget true color. If the gadgets were gadget true color and gadget true for all clauses were false, it was not 3SAT, it would have been satisfiable. So $G \notin 3COL$.

QED: G is constructible in polytime, few for loops over Φ.

We see $3SAT \leq_p 3COL$ since $3SAT$ is NP-complete and $3COL$ is NP-complete. As NP is closed under $\leq_p$, $3COL$ must also be NP-complete.

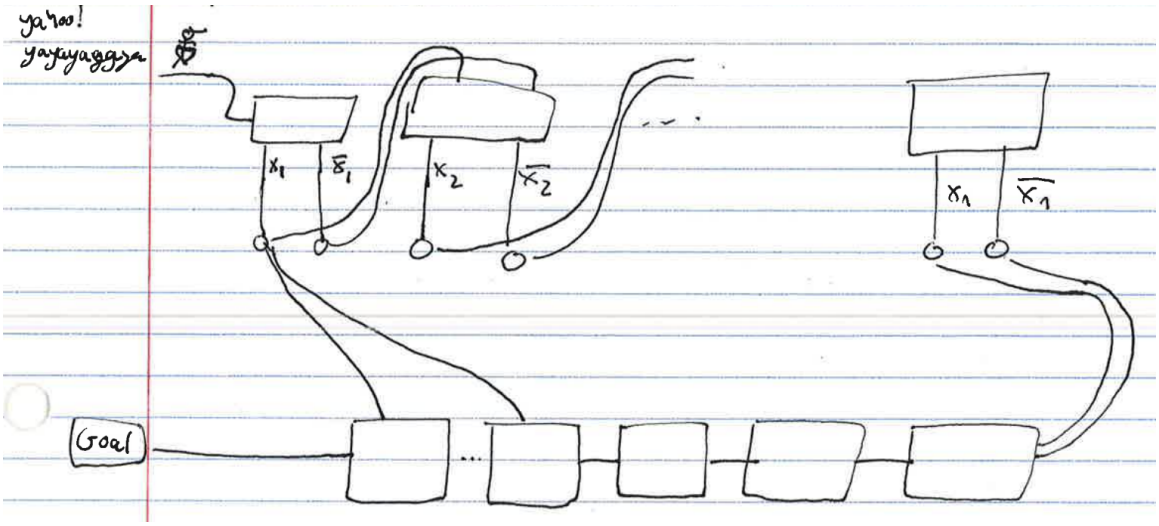
List of NP-complete games:

- Mario (probably not in NP)
- Battleship
- Sudoku on a $n \times n$ grid with n blocks
- Inverse (NP-hard, probably not in NP)
- Minesweeper

- Mastermind
- Freecell Solitaire
- Tetris

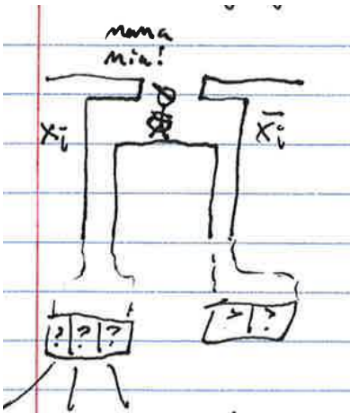
NP-completeness of Mario

Let's prove Mario is NP-complete. This comes from a great paper series by Erik Demaine. I have recently been told of a good video explanation of this reduction as well. Seems a Mario SAT NP-hard or something on YouTube. We prove 3SAT $\leq_p$ MARIO. We construct our level (that is, we prove 3SAT $\leq_p$ MARIO).

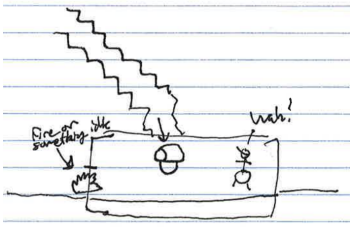


Basically, each variable gadget can only choose one of x_i or $\neg x_i$. Then we'll feed a mushroom into each clause gadget, hence can only go from each one clause to the next if we have a mushroom.

Variable Gadget



He can only choose one, but he needs mushrooms.



Needs at least one mushroom to get through the fire plant.

If Φ is 3SAT, then the level is satisfiable, Mario takes the choices according to the exact assignment of the variables. So we see L $\in$ MARIO.

If Φ is not 3SAT, some clause is always unsatisfied. Mario will get stuck in this clause gadget forever and die with no mushroom, so we see L $\notin$ MARIO.

Thus, Φ is 3SAT $\Leftrightarrow$ L $\in$ MARIO. This transformation takes polynomial time so we observe 3SAT $\leq_p$ MARIO and that Mario is NP-hard.

Why is this formulation of Mario not in NP-complete? Why is Mario not in NP? Turns out this formulation of any NP-hard is not known to be NP-complete. While it is seductive to consider a witness for a verifier as a sequence of button pushes, the verifier runs in polytime only iff the witness is in P is polynomially sized, which we don't know is true.