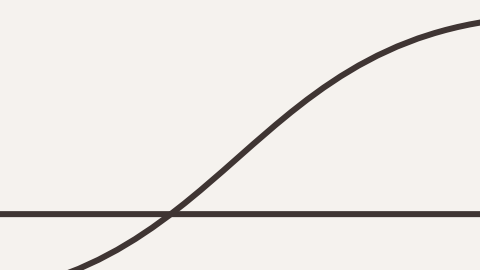


Solving Sudoku with the Power of Graph Theory

By Henry Burke and Angela Sullivan

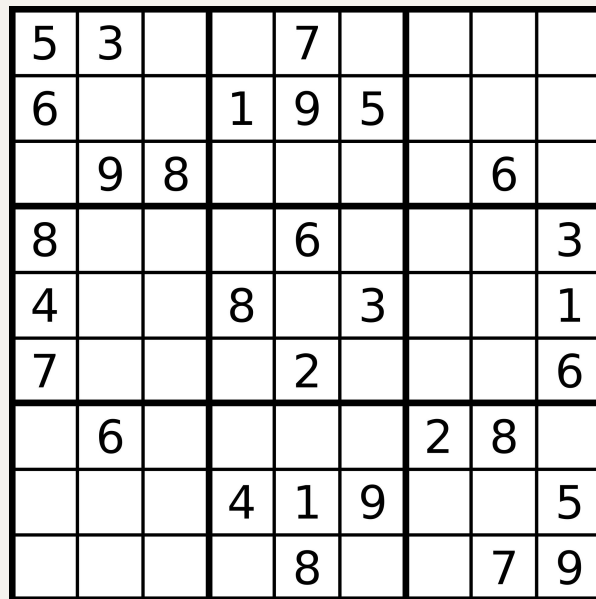


What is Sudoku???

Sudoku is a math based puzzle most often placed on a 9x9 frame work of squares

The goal is to fill all 81 squares while adhering all of the following rules

- Only numbers 1 - 9 are legal
- No vertical or horizontal rows may share a number
- No 3x3 partition may share a number



5	3			7				
6			1	9	5			
	9	8					6	
8				6				3
4			8		3			1
7				2				6
	6					2	8	
			4	1	9			5
				8			7	9

What is Sudoku???

- Sudoku is based off the the mathematical concept of a Latin Square.
- A Latin Square is an $n \times n$ matrix filled with n different symbols, where each of the n symbols occur exactly once in each row and column.

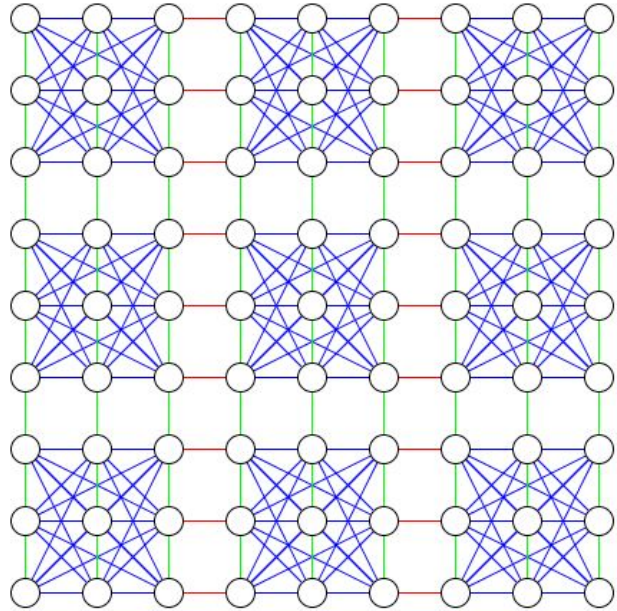


1	2	3	4	5
2	1	5	3	4
3	4	2	5	1
4	5	1	2	3
5	3	4	1	2

What is Sudoku???

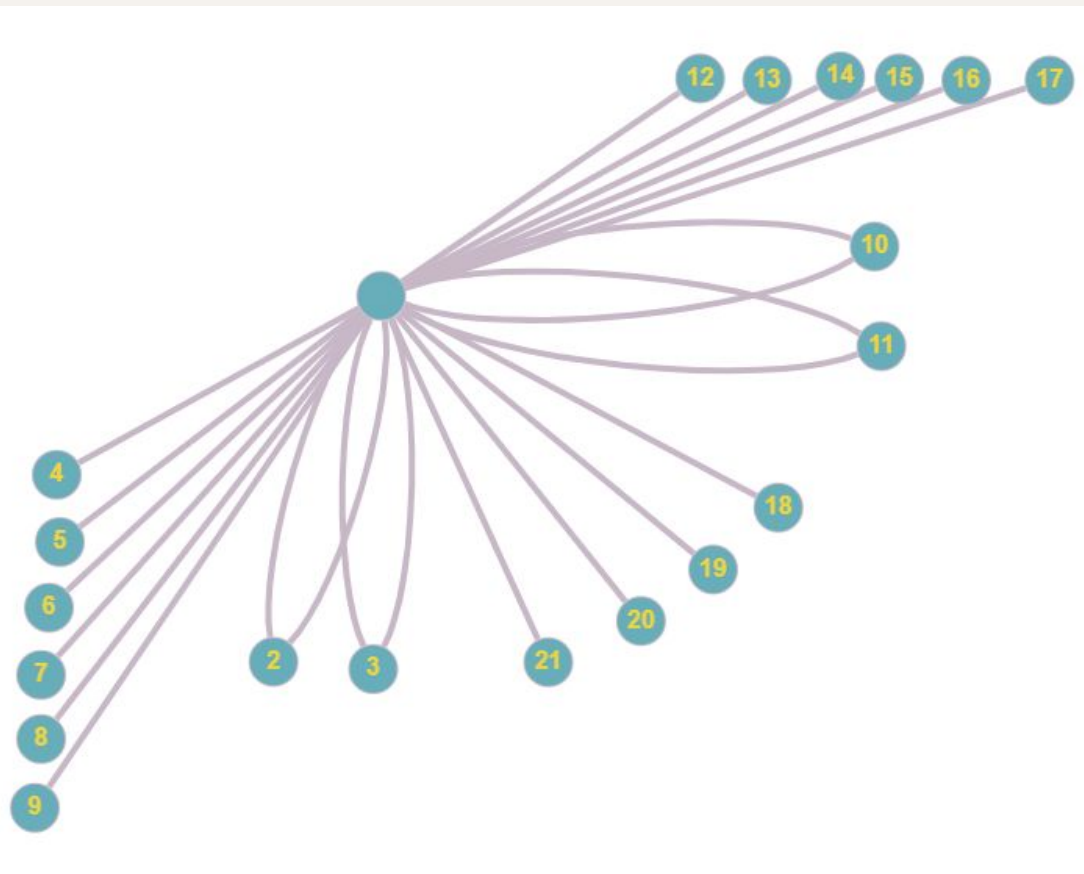
- Because a Latin Square is essentially sudoku minus the sub-squares, given an $n^2 \times n^2$ Sudoku board, it is possible to find an $n \times n$ Latin Square from that board. [Keep this in mind for later ;)]

3	9	1	2	8	6	5	7	4
4	8	7	3	5	9	1	2	6
6	5	2	7	1	4	8	3	9
8	7	5	4	3	1	6	9	2
2	1	3	9	6	7	4	8	5
9	6	4	5	2	8	7	1	3
1	4	9	6	7	3	2	5	8
5	3	8	1	4	2	9	6	7
7	2	6	8	9	5	3	4	1



Due to the nature of Sudoku we can represent the puzzle as a graph easily

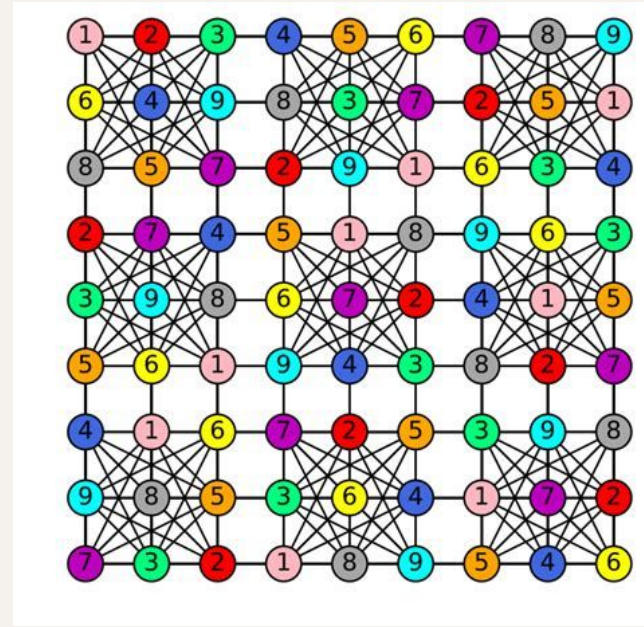
This graph shows us the connectivity but is not great for showing what connection looks like for a single node



Now what?

Representing Sudoku as a graph
lets us use graph coloring
techniques to solve incredibly
efficiently

- Every square is represented as a tuple of length two: (x, y) where $x, y \in \{1, 2, 3, 4, 5, 6, 7, 8, 9\}$
- Every vertex is of degree 24 and it will have 20 unique neighbors
- The Graph's initial state is partially colored, otherwise the coloring would be ambiguous
- Sudoku is fundamentally a minimal graph coloring problem where $k = 9$



$$\chi(G) = 9$$

Lets color!!!

1. We treat the list like a two-dimensional array and iterate through it linearly
2. We check each squares remaining possible numbers, if there is only a single number that works, we set the value and continue
3. If the there exist no squares with one unique coloring, then we will choose a square with the smallest number of potential colors and (greedily) choose the lowest number. If this number leads to and invalid solution, then we backtrack to the board prior to choosing the invalid color and choose the next possible coloring.
4. When this algorithm is finished running our sudoku will be solved // Our Graph will be minimally colored with a chromatic number of 9

Our initial board state

7				5	8			2
6		5			2			9
2		9		7			3	8
	9	7	8		3	2	5	
			2				1	3
3	4		1	9		6	8	
		3					9	
8				3		7		
	2	6	7	8		3		

We have plenty of hints so iteratively coloring should be a breeze

Our board after one iteration

This is what the board looks like after the first iteration of our lovely coloring algorithm

7				5	8			2
6		5			2			9
2	1	9		7			3	8
1	9	7	8		3	2	5	4
5		8	2			9	1	3
3	4	2	1	9		6	8	7
4		3					9	
8	5	1		3		7		6
9	2	6	7	8		3	4	

Our final board state

7	3	4	9	5	8	1	6	2
6	8	5	3	1	2	4	7	9
2	1	9	6	7	4	5	3	8
1	9	7	8	6	3	2	5	4
5	6	8	2	4	7	9	1	3
3	4	2	1	9	5	6	8	7
4	7	3	5	2	6	8	9	1
8	5	1	4	3	9	7	2	6
9	2	6	7	8	1	3	4	5

As you can see the puzzle is solved, what you can't see is that this is the fifth iteration of our simple coloring algorithm.

When compared to a basic recursive backtracking algorithm, coloring blows it away.

Our competition

Compared to our graph coloring algorithm, recursive backtracking makes many more comparisons.

Most of the comparisons backtracking makes end up being without purpose. It just brute force

5	3	1	2	7	6	8	9	4
6	2	4	1	9	5	2		
	9	8					6	
8				6				3
4			8		3			1
7				2				6
	6					2	8	
			4	1	9			5
				8			7	9

Code Pumpkin (1)

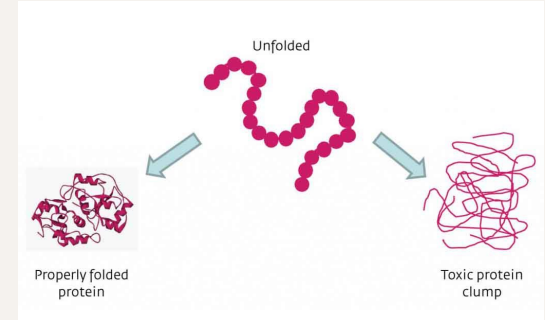
Practical Application of Sudoku

Studying how to solve sudoku is useful in understanding NP-complete problems

Nondeterministic Polynomial time (NP) is the set of problems, which can be checked in polynomial time, but take exponential time to solve.

Since all NP-complete problems can be reduced or transformed to form each other, by understanding Sudoku we understand all other NP-complete problems by proxy

Examples of NP complete problems (besides sudoku):
job scheduling, protein folding, circuit design,
Battleship, assembling an optimal Bitcoin block



Sudoku NP-Complete Explanation

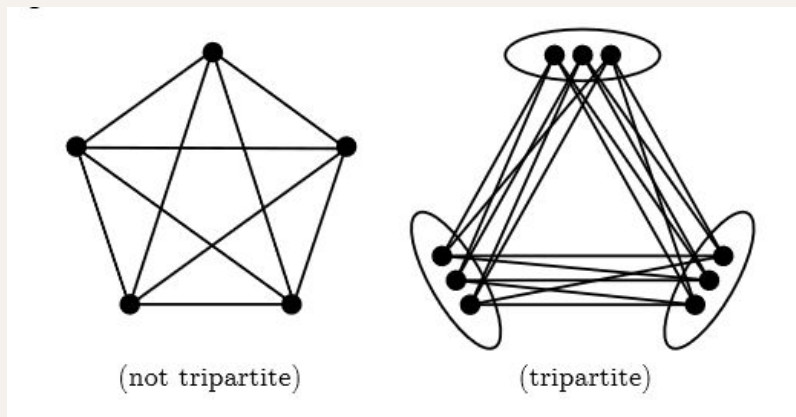
- To show that Sudoku is NP-Complete we must show that Sudoku reduces to a Latin Square which reduces to a triangulated tripartite graph which reduces to a 3SAT, which is known to be NP-Complete.
 - Sudoku > Latin Square > Triangulated Tripartite Graph > 3SAT

Sudoku NP-Complete Explanation

- To show that Sudoku is NP-Complete we must show that Sudoku reduces to a Latin Square which reduces to a triangulated tripartite graph which reduces to a 3SAT, which is known to be NP-Complete, which is known to be NP-Complete.
 - **Sudoku > Latin Square** > Triangulated Tripartite Graph > 3SAT
- Because a Latin Square is essentially sudoku minus the sub-squares, given an $n^2 \times n^2$ Sudoku board, it is possible to find an $n \times n$ Latin Square from that board. Therefore, Sudoku can be reduced to a Latin Squares problem.

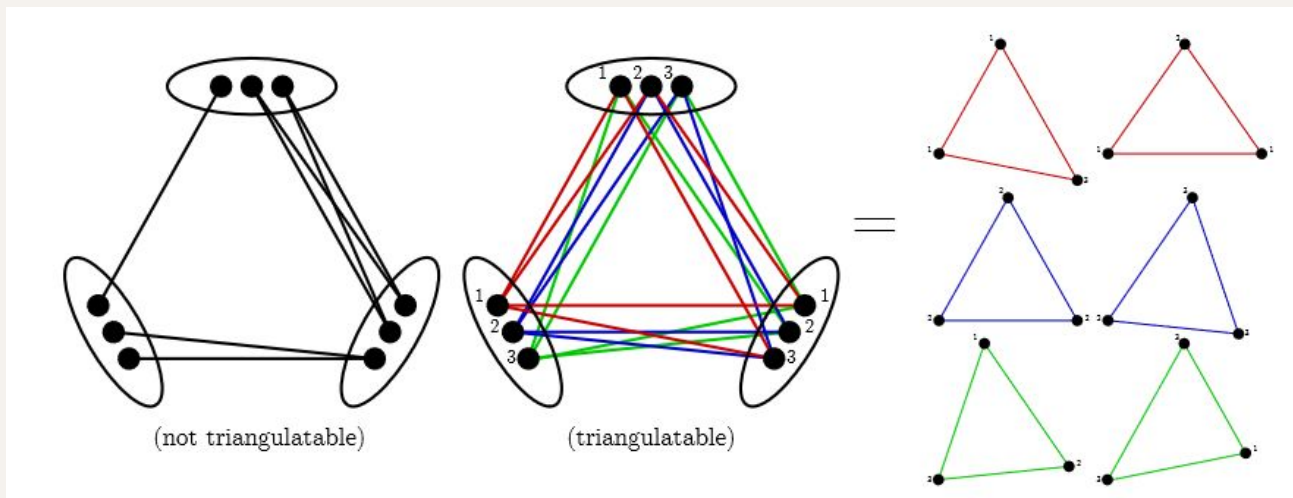
Sudoku NP-Complete Explanation

- To show that Sudoku is NP-Complete we must show that Sudoku reduces to a Latin Square which reduces to a triangulated tripartite graph which reduces to a 3SAT, which is known to be NP-Complete, which is known to be NP-Complete.
- - Sudoku > **Latin Square** > **Triangulated Tripartite Graph** > 3SAT



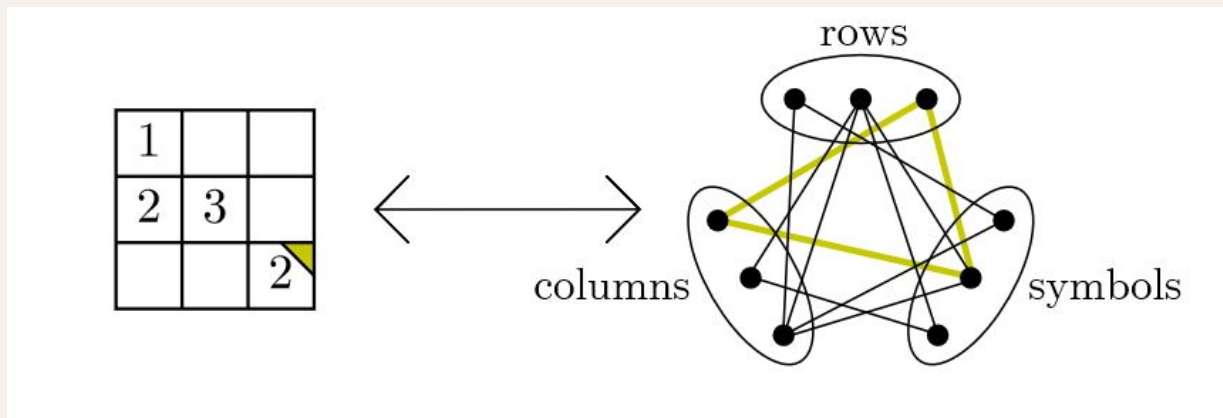
Sudoku NP-Complete Explanation

- To show that Sudoku is NP-Complete we must show that Sudoku reduces to a Latin Square which reduces to a triangulated tripartite graph which reduces to a 3SAT, which is known to be NP-Complete.
 - Sudoku > **Latin Square** > **Triangulated Tripartite Graph** > 3SAT



Sudoku NP-Complete Explanation

- To show that Sudoku is NP-Complete we must show that Sudoku reduces to a Latin Square which reduces to a triangulated tripartite graph which reduces to a 3SAT, which is known to be NP-Complete.
 - Sudoku > **Latin Square** > **Triangulated Tripartite Graph** > 3SAT

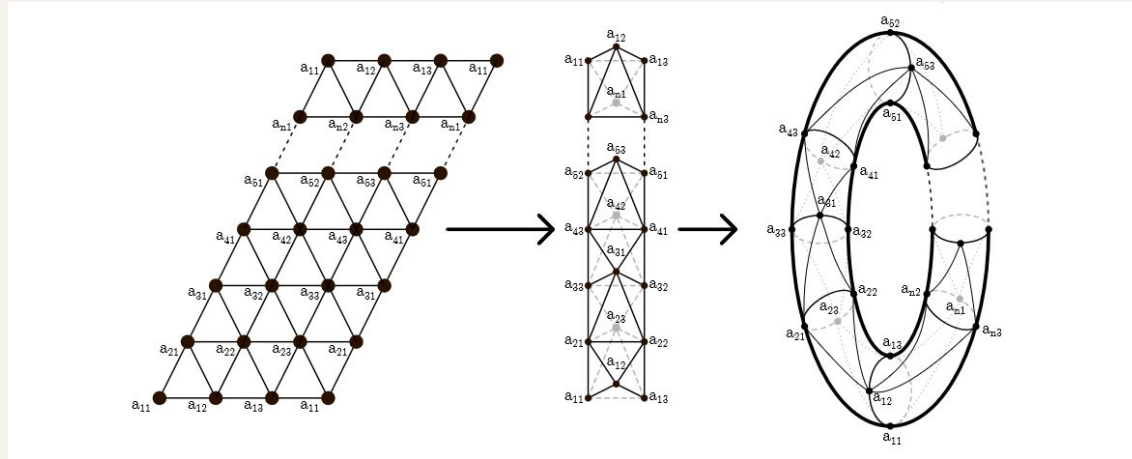


Sudoku NP-Complete Explanation

- To show that Sudoku is NP-Complete we must show that Sudoku reduces to a Latin Square which reduces to a triangulated tripartite graph which reduces to a 3SAT, which is known to be NP-Complete.
 - Sudoku > Latin Square > **Triangulated Tripartite Graph > 3SAT.**
- The SAT or Boolean satisfiability problem is a problem that asks how fast can it be to tell if a given formula in Boolean algebra with an unknown number of variables is solvable.
- A problem is solvable if there exists some combination of binary values (0 and 1) that gives us 1.
- The 3SAT is only concerned with asking how fast is it to check if a given formula in boolean algebra with 3 variables is solvable

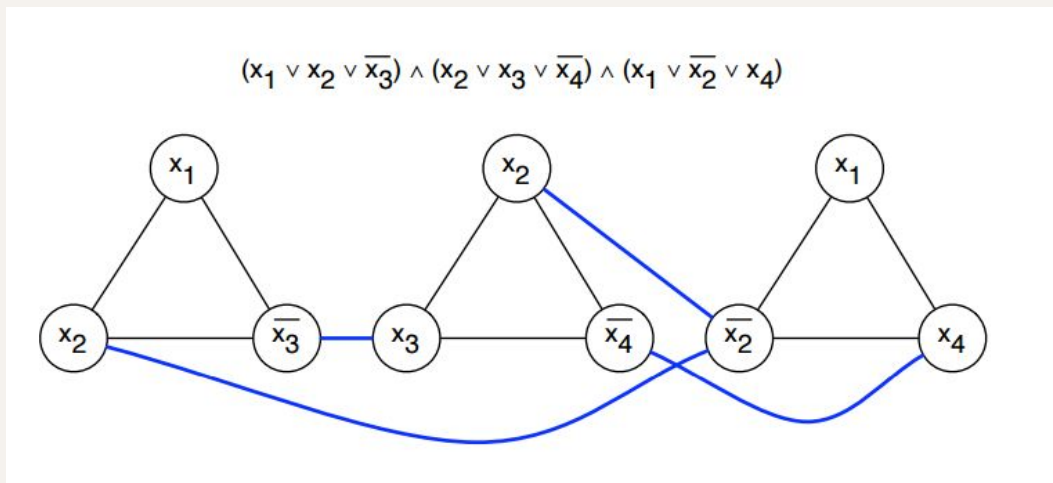
Sudoku NP-Complete Explanation

- To show that Sudoku is NP-Complete we must show that Sudoku reduces to a Latin Square which reduces to a triangulated tripartite graph which reduces to a 3SAT, which is known to be NP-Complete.
 - Sudoku > Latin Square > **Triangulated Tripartite Graph** > **3SAT**.



Sudoku NP-Complete Explanation

- To show that Sudoku is NP-Complete we must show that Sudoku reduces to a Latin Square which reduces to a triangulated tripartite graph which reduces to a 3SAT, which is known to be NP-Complete.
 - Sudoku > Latin Square > **Triangulated Tripartite Graph** > **3SAT**.



Conclusion

- Sudoku is a fun game that involves a surprising amount of graph theory
- Sudoku can be only (currently) be solved in exponential time, but can be checked in quadratic time.
- Understood how to solve sudoku can help us to solve many other problems within mathematics and other fields.



Citations

- Andhariya, A. (2018, October 22). *Sudoku solver using recursive backtracking*. Code Pumpkin. <https://codepumpkin.com/sudoku-solver-using-backtracking/>
- Bartlett, P. (2014). Minilecture 7: 3SAT and Latin Squares. Retrieved April 30, 2023, from http://web.math.ucsb.edu/~padraic/ucsb_2013_14/mathcs103_s2014/mathcs103_s2014_mlecture7.pdf
- Goldberg, P. (n.d.). *NP-completeness of sudoku*. MSc course: Foundations of Computer Science. Retrieved April 30, 2023, from <http://www.cs.ox.ac.uk/people/paul.goldberg/FCS/sudoku.html>
- Ru, R., Huang, K., & Yu, L. N. (n.d.). *Discussing NP-Completeness of Sudoku Game*. Discussing NP-completeness of Sudoku game. Retrieved April 30, 2023, from <https://jcrouser.github.io/CSC250/projects/sudoku.html>
- Samarzija, T. (n.d.). *3SAT*. Brilliant Math & Science Wiki. Retrieved April 30, 2023, from <https://brilliant.org/wiki/3sat/>
- Weisstein, E. W. (n.d.). *Latin square*. from Wolfram MathWorld. Retrieved April 30, 2023, from <https://mathworld.wolfram.com/LatinSquare.html>
- Yato, T. (2003). COMPLEXITY AND COMPLETENESS OF FINDING ANOTHER SOLUTION AND ITS APPLICATION TO PUZZLES [Master's thesis, University of Tokyo]. Retrieved April 30, 2023, from <https://www.imai.is.s.u-tokyo.ac.jp/~yato/data2/MasterThesis.pdf>

DEMO!!!!