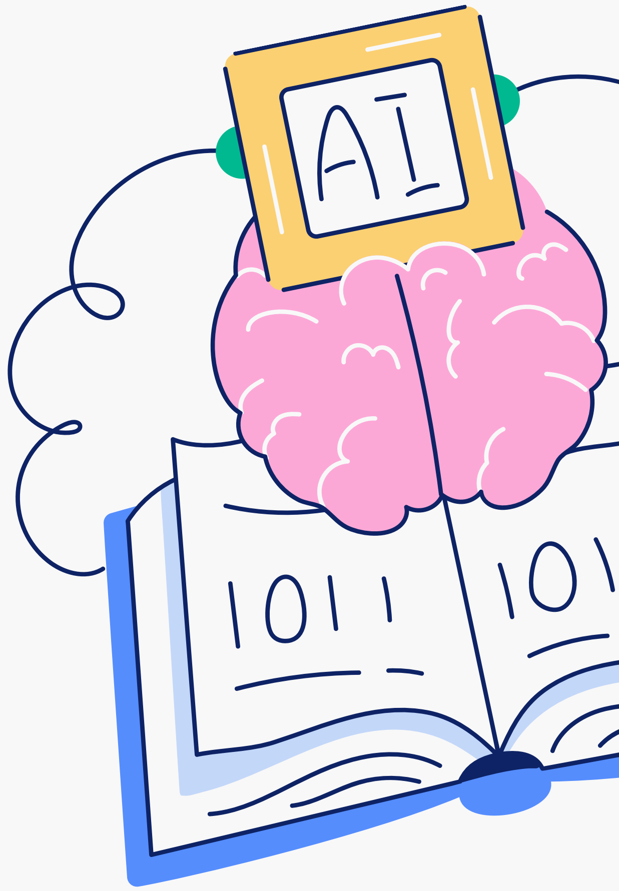
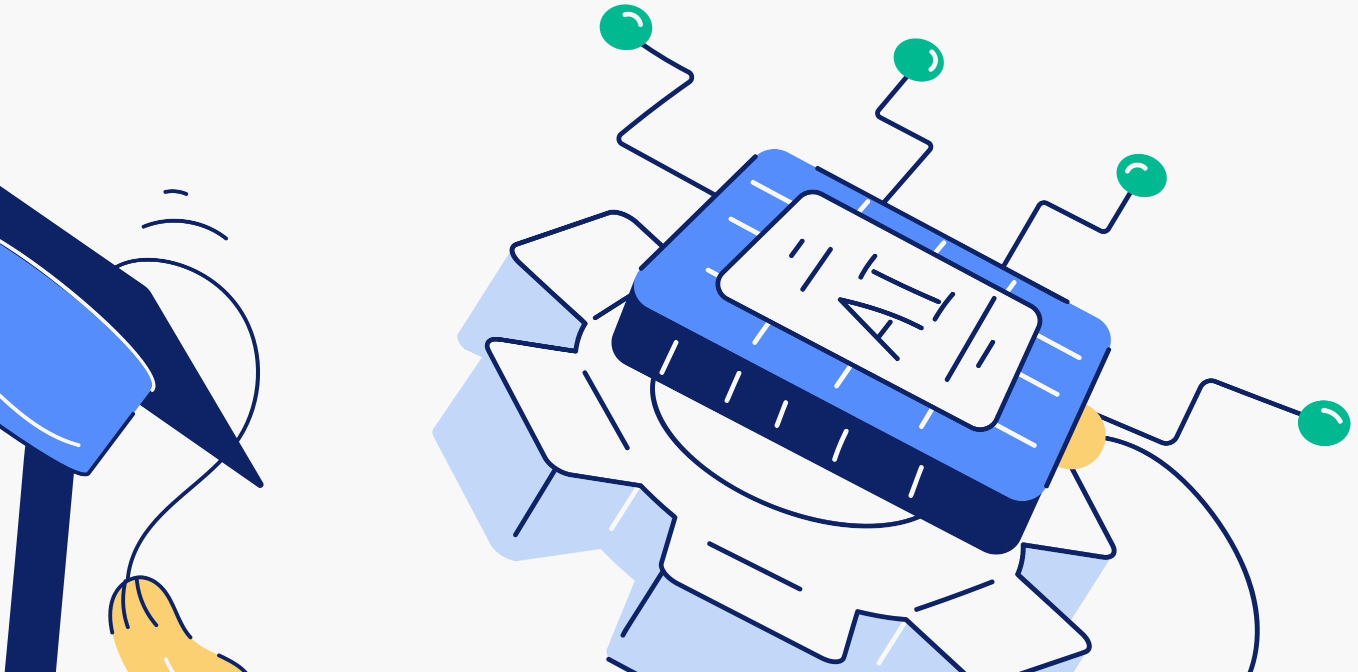
Course : CSCI 331 - Introduction to Artificial Intelligence
Rochester Institute of Technology
December 3, 2025

Garett (Gayle) Rogers, Veronika Kirievskaya, Mate Revishvili

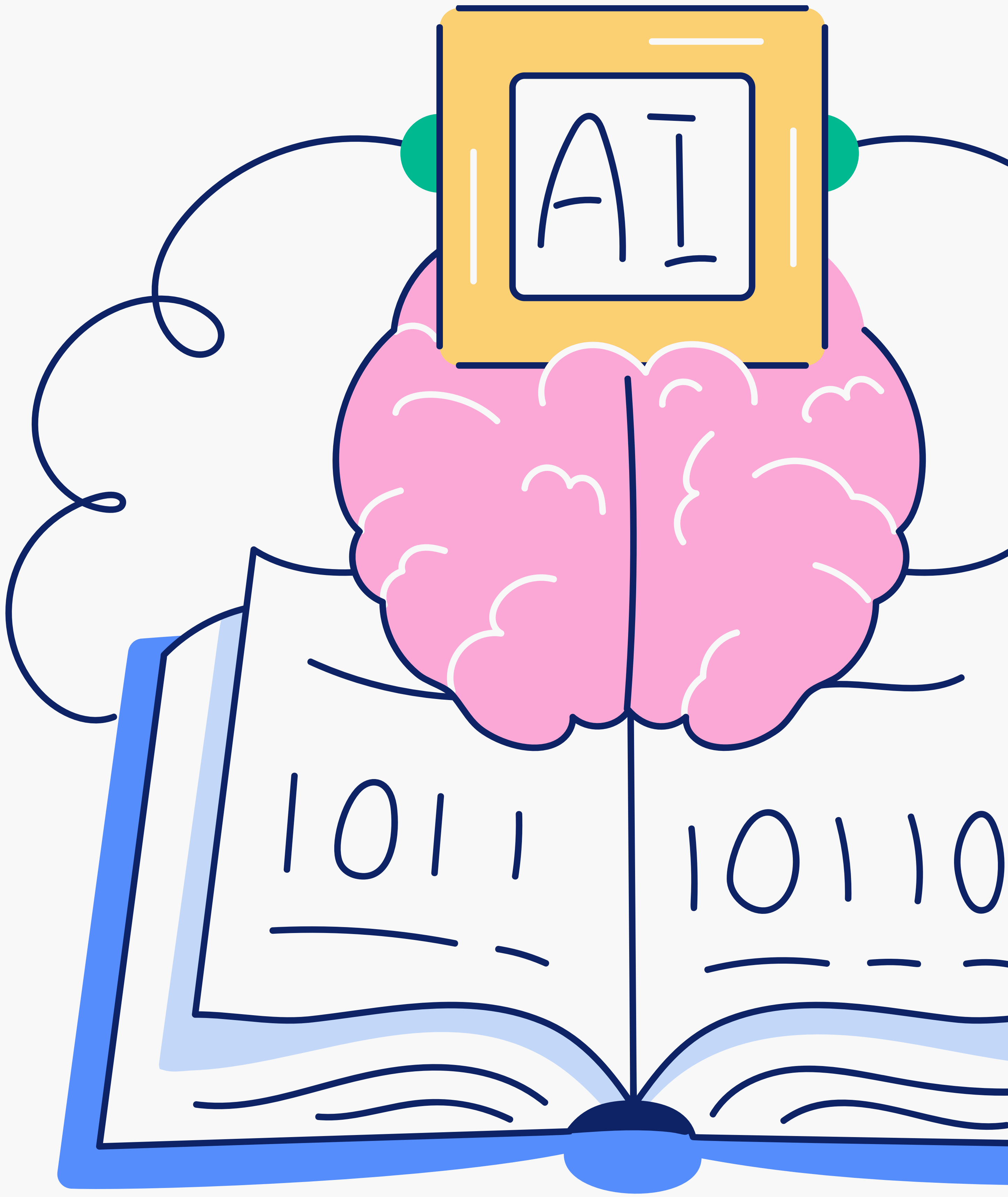
CSP SOLVER FOR CUSTOMIZED SUDOKU PUZZLES





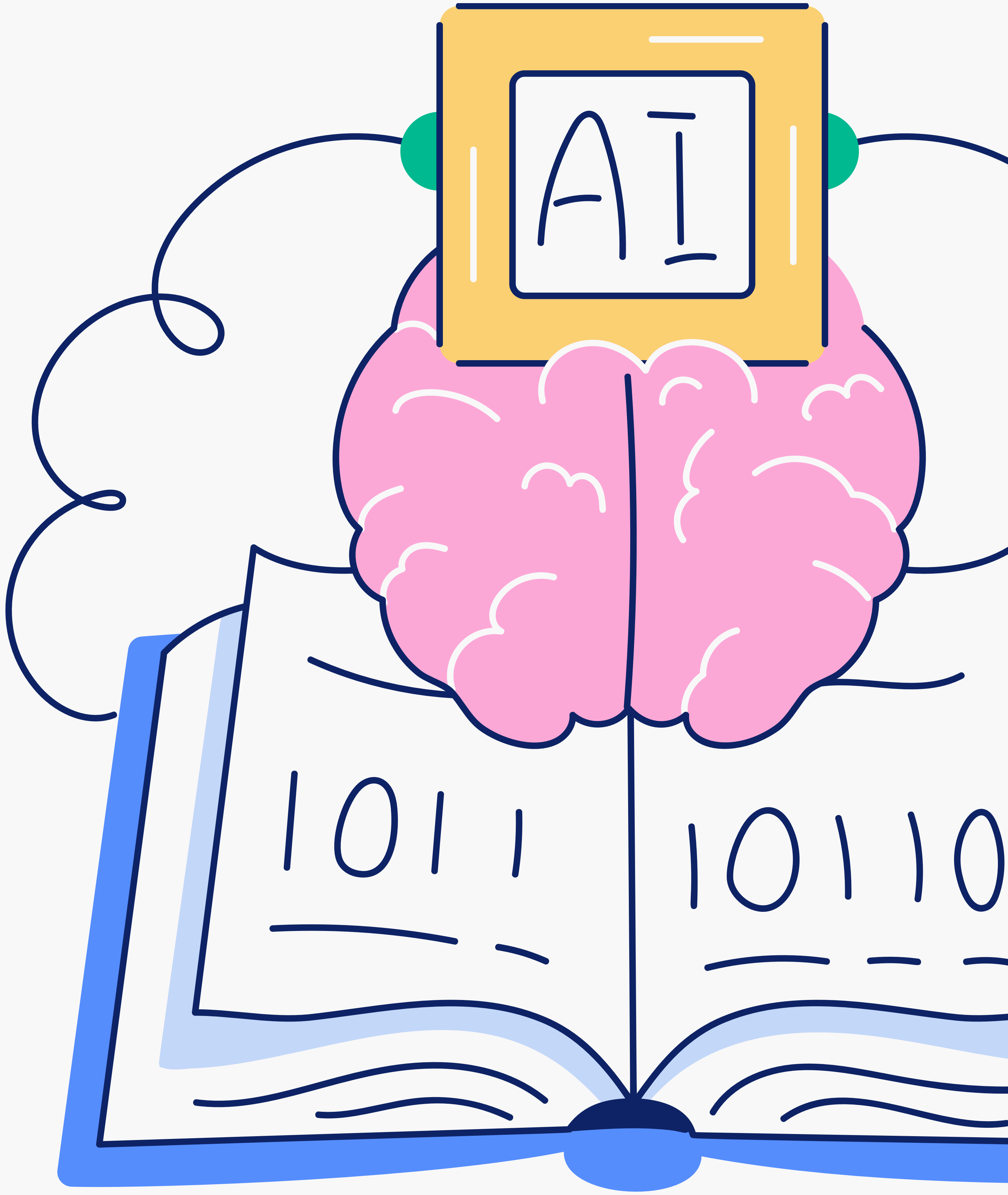
PROBLEM

- Implement an efficient solver for Sudoku Puzzles using constraint satisfaction techniques.
- Contraints are the rules of Sudoku
    - Each row and column must all have unique digits
    - Each section (square, of length SQRT(length)), also has unique digits



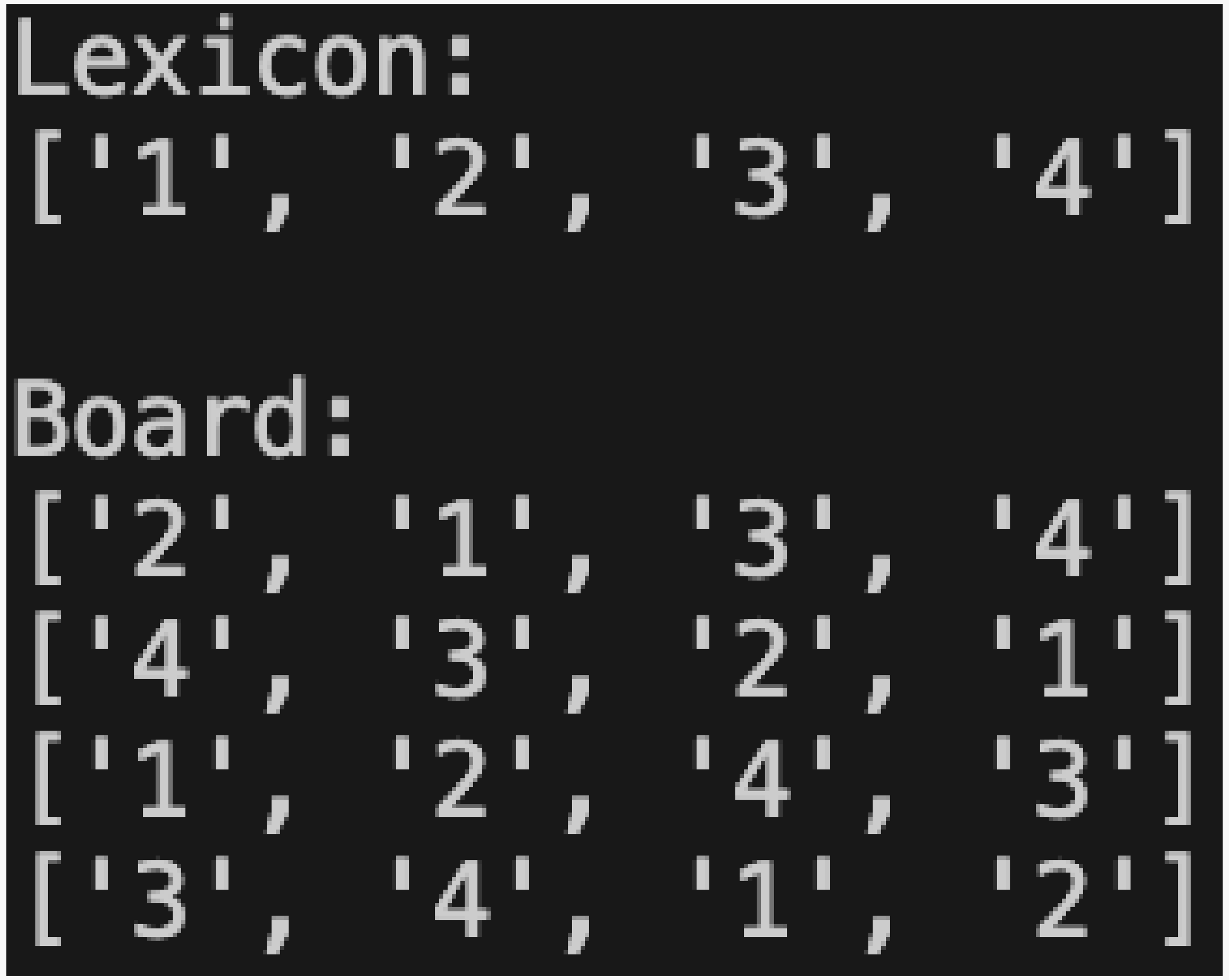
HYPOTHESIS

- Difference in efficiency between the two algorithms?
    - We expect the CSP-enhanced version to perform better
- Why?
    - Forward checking reduces the number of dead ends explored, so the correct solution is found faster



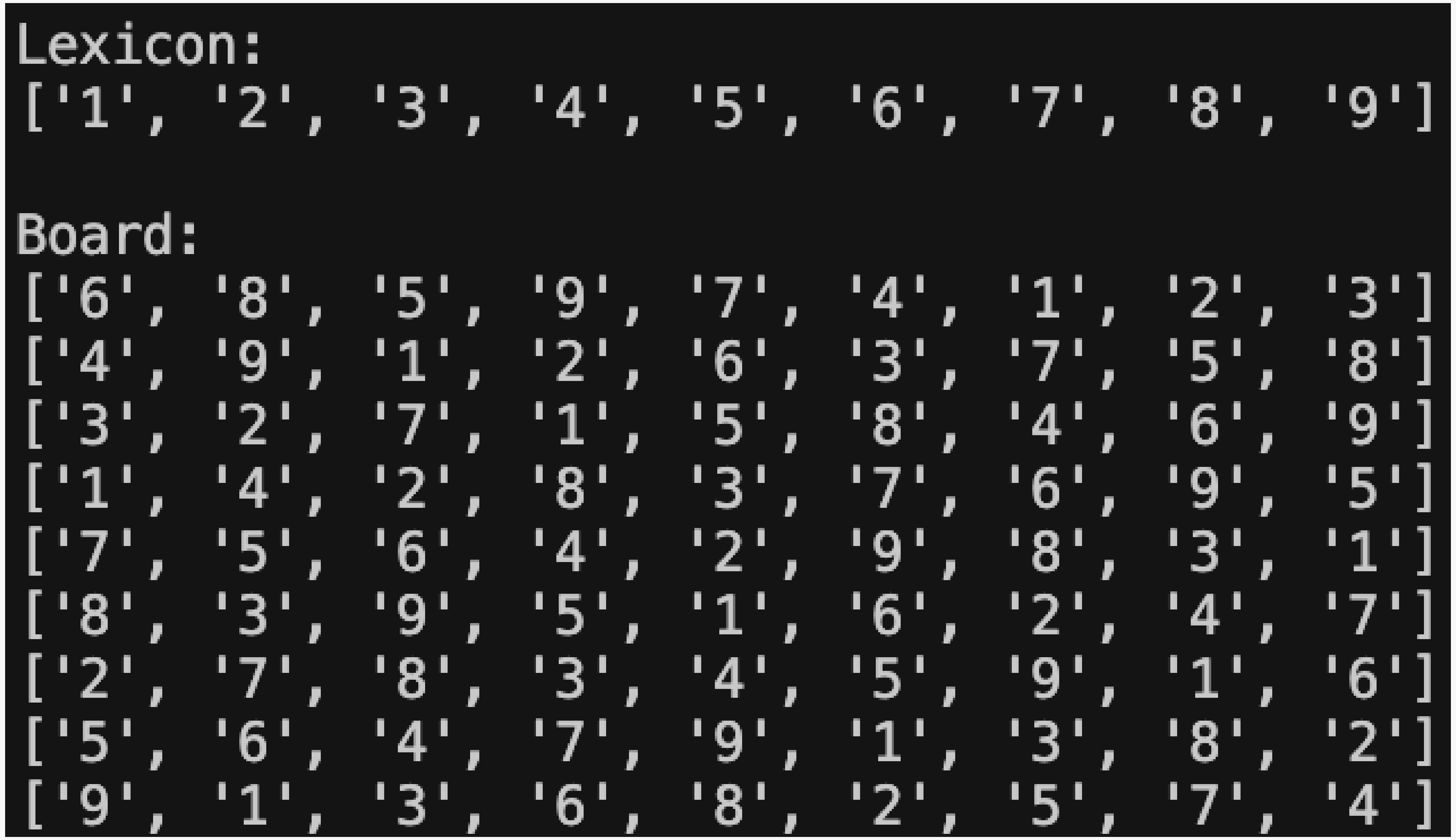
EXAMPLE SUDOKU BOARDS

We tested our CSP solver on sudoku boards of size 4x4 and size 9x9. Boards of size 16x16 were too large to run in a reasonable amount of time.



```
Lexicon:
['1', '2', '3', '4']

Board:
['2', '1', '3', '4']
['4', '3', '2', '1']
['1', '2', '4', '3']
['3', '4', '1', '2']
```



```
Lexicon:
['1', '2', '3', '4', '5', '6', '7', '8', '9']

Board:
['6', '8', '5', '9', '7', '4', '1', '2', '3']
['4', '9', '1', '2', '6', '3', '7', '5', '8']
['3', '2', '7', '1', '5', '8', '4', '6', '9']
['1', '4', '2', '8', '3', '7', '6', '9', '5']
['7', '5', '6', '4', '2', '9', '8', '3', '1']
['8', '3', '9', '5', '1', '6', '2', '4', '7']
['2', '7', '8', '3', '4', '5', '9', '1', '6']
['5', '6', '4', '7', '9', '1', '3', '8', '2']
['9', '1', '3', '6', '8', '2', '5', '7', '4']
```

METHODS

Algorithms implemented:

- Plain backtracking
- Pruned backtracking (validity checks)
- Forward checking (domain pruning)

The plain backtracking (without validity checks) takes too long to produce useful results (hours for some graphs).

METHODS

Design choices:

- Board class to build a standard representation of a sudoku board given a lexicon and file or user input.
- PTUI for users to easily run small test cases or to test the csp algorithms on sudoku boards.
- Test boards of two sizes : 4×4 and 9×9

Evaluation metrics:

1) We checked that the algorithms found a valid solution to the sudoku puzzle.

2) Runtime (in seconds)

3) Backtrack count - number of backtracking steps that the algorithm took

TEST RESULTS

Normal backtracking, without any pruning, would run slowly (in comparison) on 4x4 boards and would run too slowly on 9x9 boards (didn't terminate within 5 minutes).

ALGORITHM	NUMBER OF BACKTRACKING STEPS	RUNTIME (IN SECONDS)
NORMAL BACKTRACKING	1032502 STEPS	1.1828 SECONDS
BACKTRACKING WITH PRUNING	0 STEPS	0.00018 SECONDS
FORWARD CHECKING	0 STEPS	0.0012 SECONDS

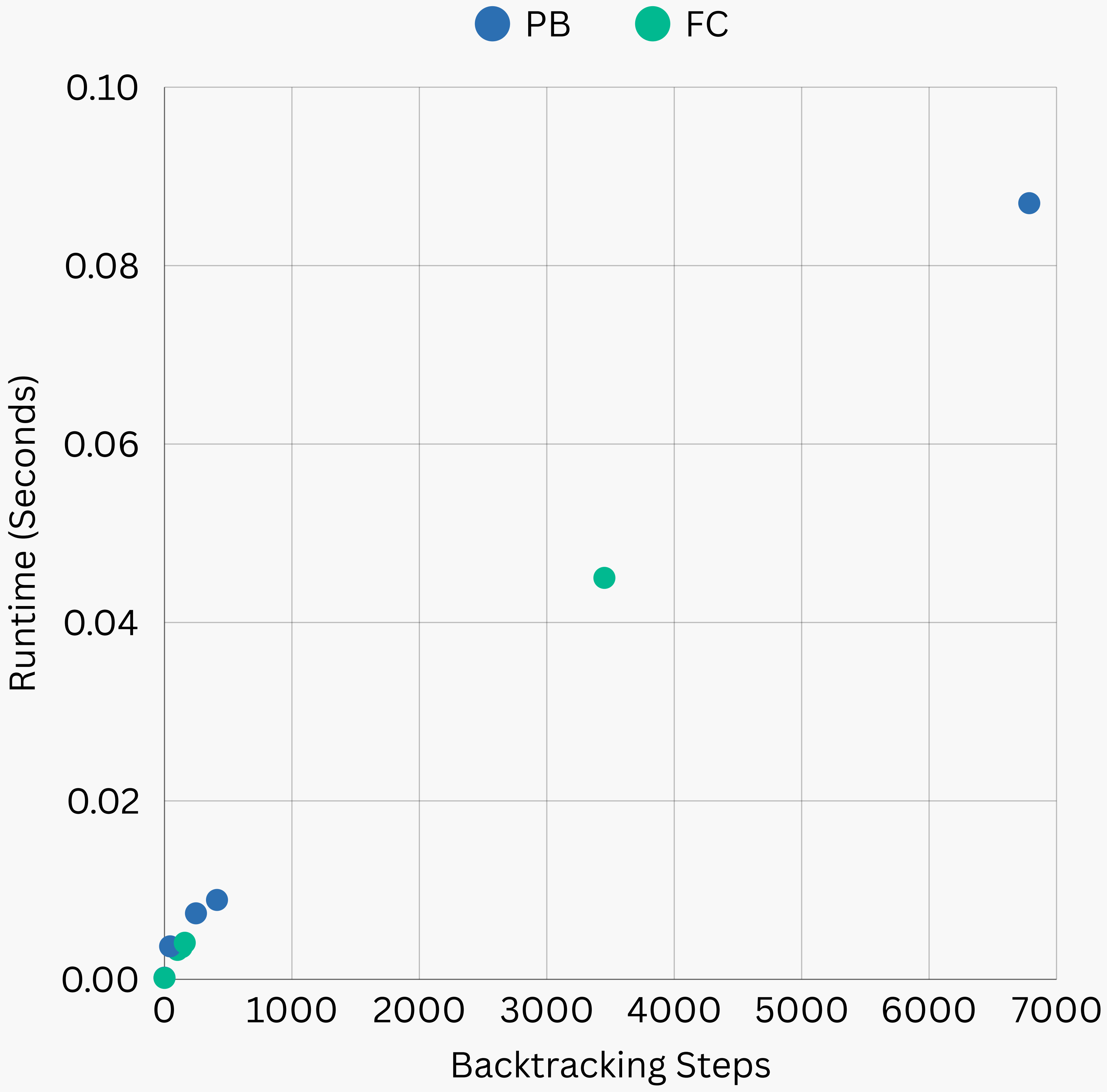
TEST RESULTS

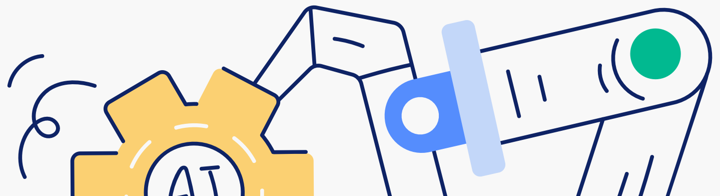
Number of backtracking steps and runtime in seconds that the backtracking with pruning and forward checking algorithms took.

	BACKTRACKING WITH PRUNING		FORWARD CHECKING	
BOARD SIZE	NUMBER OF BACKTRACKING STEPS	RUNTIME	NUMBER OF BACKTRACKING STEPS	RUNTIME
4X4	0	0.00018	0	0.0012
9X9	412	0.0089	135	0.0036
9X9	247	0.0074	102	0.0033
9X9	44	0.0038	159	0.0041
9X9	6786	0.085	3452	0.026

RESULTS

The plain backtracking with pruning algorithm took both more seconds of runtime and more backtracking steps.





REFLECTION

- Backtracks vs. Runtime:
  - Forward checking (when implemented efficiently) yielded significant improvements in both the speed of the algorithm as well as the number of backtracking steps
- Scalability:
  - CSP has its limits, with runtime blowing up for larger boards, like one of size 16×16.

QUESTIONS?