AI Homework 6

Ando Pepe, Patrick Gardner, Zachary Vega

Implementation:

For this assignment we implemented a backtracking algorithm with forward checking that is able to solve Sudoku puzzles. The implementation is divided into two main parts. Part 1 consists of functions that support solving the sudoku puzzle. Part 2 is a class that supports displaying the state of the Sudoku puzzle. The sudoku solver uses backtracking with a heuristic to fill out the grid. The implementation is broken up into different functions:

solve_sudoku – This function is a wrapper around the solve() function and is passed a 2d array with zero's indicating empty spaces and integers 1-9 indicating the spots that are filled.

is_valid – This helper function determines if a specific number can go in a specific grid position. It checks horizontally, vertically, and in the corresponding 3x3 box.

find_empty – Returns an empty space in the board that still exist or None if the board is full

domain_size – This helper function determines the size of domain (number of possible values that can go into the specific position). If the domain is smaller than the corresponding grid position is more constrained.

degree – This helper function determines the degree of a specific position in the grid. This helps select cells that constrain other cells the most, breaking ties when multiple cells have the same domain size.

solve – This is the recursive backtracking function that optimizes solving the puzzle.

Through the use of these functions, we were successfully able to use backtracking to get fast and correct results for all 3 boards that were given to us.

For additional information on classes, methods, and functions refer to additional comments in the code.

System Requirements

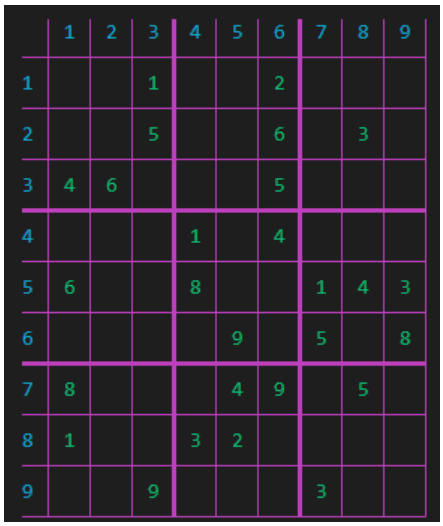
Language: Python 3.12

The following results were produced on a laptop with an “Intel i7 evo” CPU with 16 GB of RAM

Results:

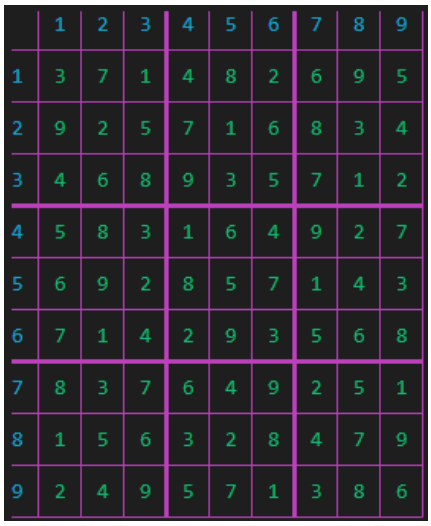
A)

Initial state



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

Result state



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

First 4 variables assignments –

Variable: (1, 1), Domain Size: 4, Degree: 19, Value: 3
Variable: (2, 1), Domain Size: 4, Degree: 18, Value: 2
Variable: (4, 1), Domain Size: 4, Degree: 20, Value: 5
Variable: (6, 1), Domain Size: 3, Degree: 19, Value: 7

Execution time = 0.078125 seconds

B)

Initial state

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

Result state

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

First 4 variables assignments –

Variable: (1, 1), Domain Size: 2, Degree: 18, Value: 3
Variable: (2, 1), Domain Size: 2, Degree: 16, Value: 6
Variable: (5, 1), Domain Size: 3, Degree: 19, Value: 8
Variable: (7, 1), Domain Size: 6, Degree: 22, Value: 4

Execution time = 2.390625 seconds

C)

Initial state

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

Result state

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

First 4 variables assignments –

Variable: (2, 1), Domain Size: 4, Degree: 15, Value: 4
Variable: (3, 1), Domain Size: 4, Degree: 20, Value: 8
Variable: (5, 1), Domain Size: 4, Degree: 21, Value: 2
Variable: (6, 1), Domain Size: 3, Degree: 18, Value: 5

Execution time = 0.125 seconds