

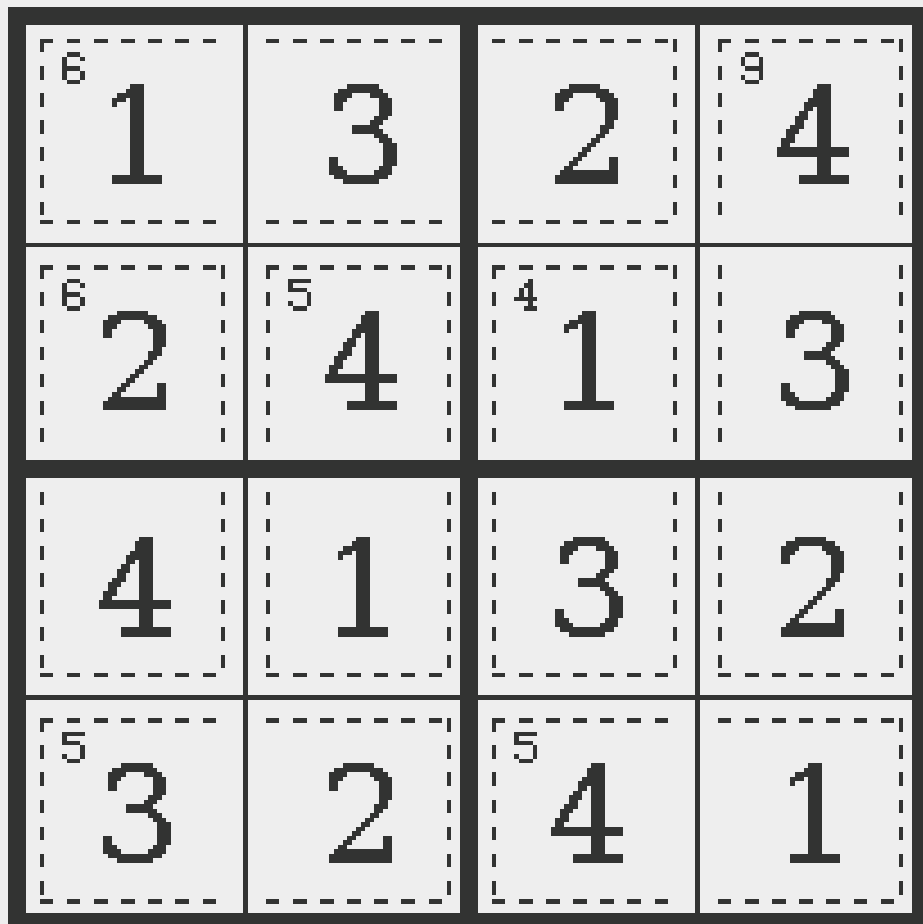


# KILLER SUDOKU

KNOWLEDGE REPRESENTATION PROJECT

TEAM 7 – MARCO BELLIZZI, DIEGO DE BARTOLO,  
DOMENICO SPAGNOLO

<https://github.com/MarcoBellizzi/Killer-Sudoku>



The image shows a 4x4 grid representing a Killer Sudoku puzzle. The grid is divided into four 2x2 quadrants by a thick black border. Each quadrant contains two cages, indicated by dashed lines. Each cage has a small number in its top-left corner representing the sum of the numbers inside. The numbers in the cages are as follows:

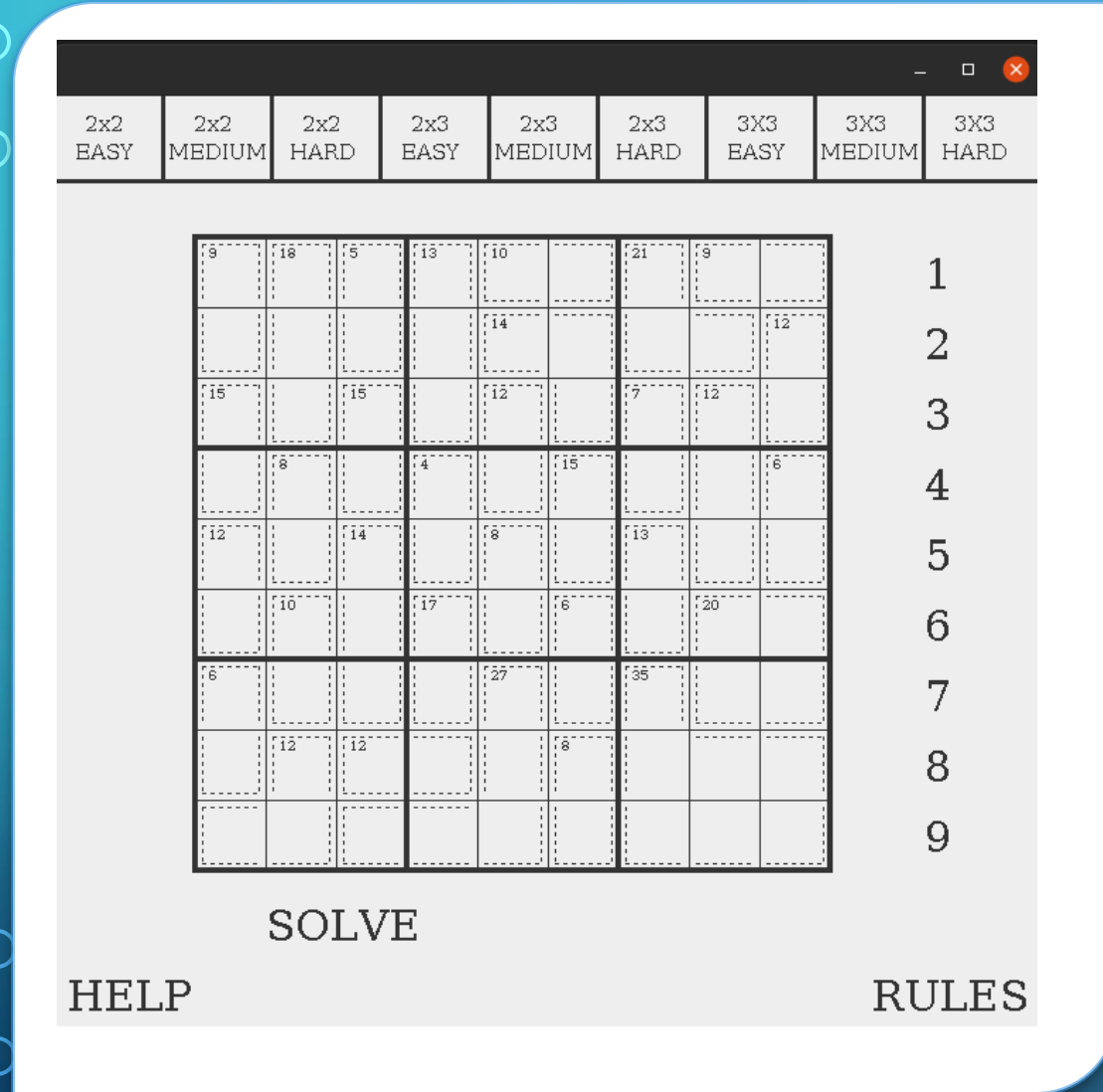
<sup>6</sup> 1	3	2	<sup>9</sup> 4
<sup>6</sup> 2	<sup>5</sup> 4	<sup>4</sup> 1	3
4	1	3	2
<sup>5</sup> 3	2	<sup>5</sup> 4	1

# KILLER SUDOKU - RULES

Killer sudoku is a puzzle that combines elements of sudoku and kakuro.

The rules are the following:

- Each row, column, and sector contains each number exactly once.
- The sum of all numbers in a cage must match the small number printed in its corner.
- No number appears more than once in a cage.



# REALIZATION

This project is made in Java, using Java Swing. The project uses Minizinc to check the solutions. It requires Internet connection to get the instances from Internet.

# MINIZINC MODEL SNEAK PEEK

```
constraint forall(i in 1..num_of_regions) ( % elements in each region are different
  forall(j in 1..regions[i, 2]) (
    alldifferent([matrix[regions[i, j*2 + 1], regions[i, j*2 + 2]]])
  )
);
```

```
constraint forall(i in 1..num_of_regions) ( % the sum of each region is equal to the sum
of each element in that region
  regions[i, 1] == sum(j in 1..regions[i, 2]) (matrix[regions[i, j*2 + 1], regions[i, j*2 + 2]]
);
```

# INPUT VALIDATION

```
constraint assert(forall(i in 1..n*m, j in 1..n*m) ( % user input validation
    matrix[i,j] > 0 /\ matrix[i,j] <= n*m
), "wrong user input value");
```

```
constraint assert(forall(i in 1..num_of_regions) ( % site input validation
    forall(j in 1..regions[i, 2]) (
        regions[i, j*2 + 1] > 0 /\ regions[i, j*2 + 2] <= n*m
    )
), "wrong site input value");
```

# INSTANCES

- `curl https://www.puzzle-killer-sudoku.com/ | grep task | sed \"s/.*task =  
'//\\\" | sed \"s/'.*//\\`
- The project retrieves the instances through the command shown above. Parse the response, extracting the information required.

# SUGGESTIONS

The GUI tells you if you do any mistakes:

- Color the row, the column, the sector or the cage in red if find at least two elements equal.
- Color the cage in red if the sum doesn't match the value on its corner.

2x2 EASY 2x2 MEDIUM 2x2 HARD 2x3 EASY 2x3 MEDIUM 2x3 HARD 3x3 EASY 3x3 MEDIUM 3x3 HARD

4	2	14		16	11	14		14	1	1
	3	13				14				2
19	7				11				10	3
	1	14		12		8	7		1	4
	16	11			9			18		5
		8		9		17	4	16		6
7	15	1	15		4					7
	18	4			9				3	8
1			13		18					9

SOLVE

HELP RULES

The background is a blue gradient with decorative white circuit-like lines in the corners. These lines consist of straight segments and small circles, resembling a stylized electronic circuit or data paths.

THANKS FOR THE ATTENTION