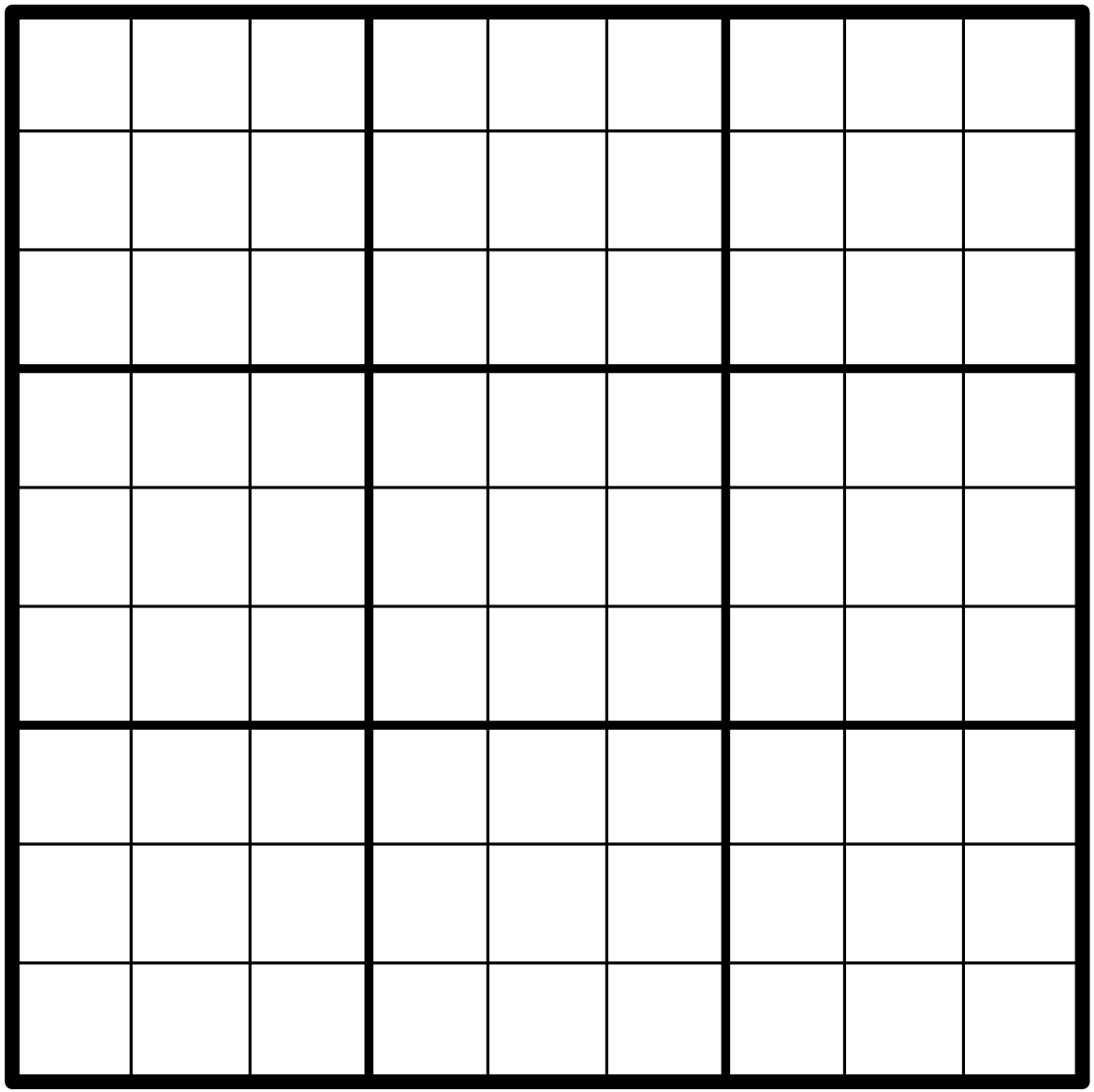
Sudoku

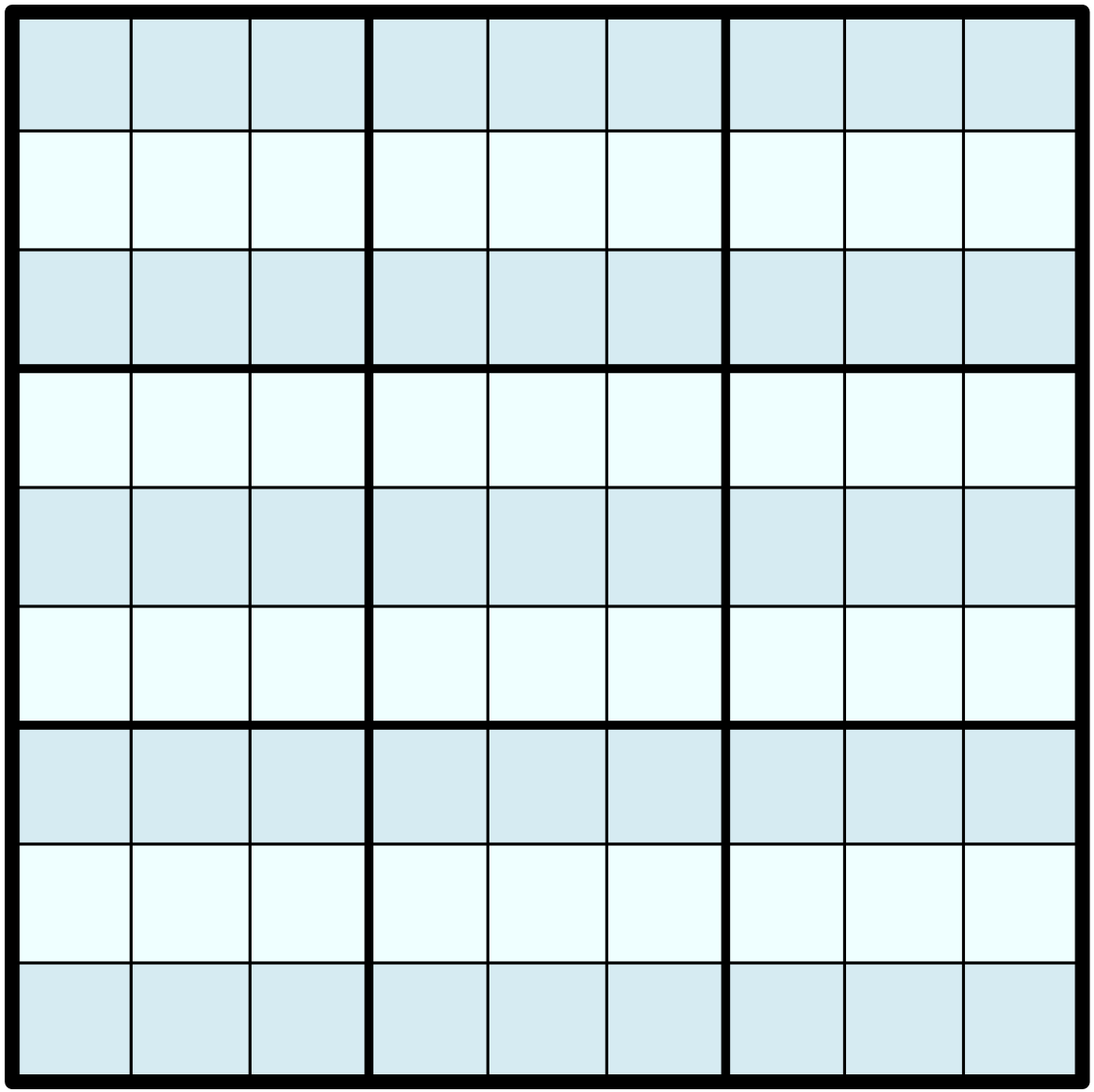
Algorithms & Games

NEW: At `https://animalgo.ist.tugraz.at` there is an animated algorithm that uses the explained approach.

Sudoku

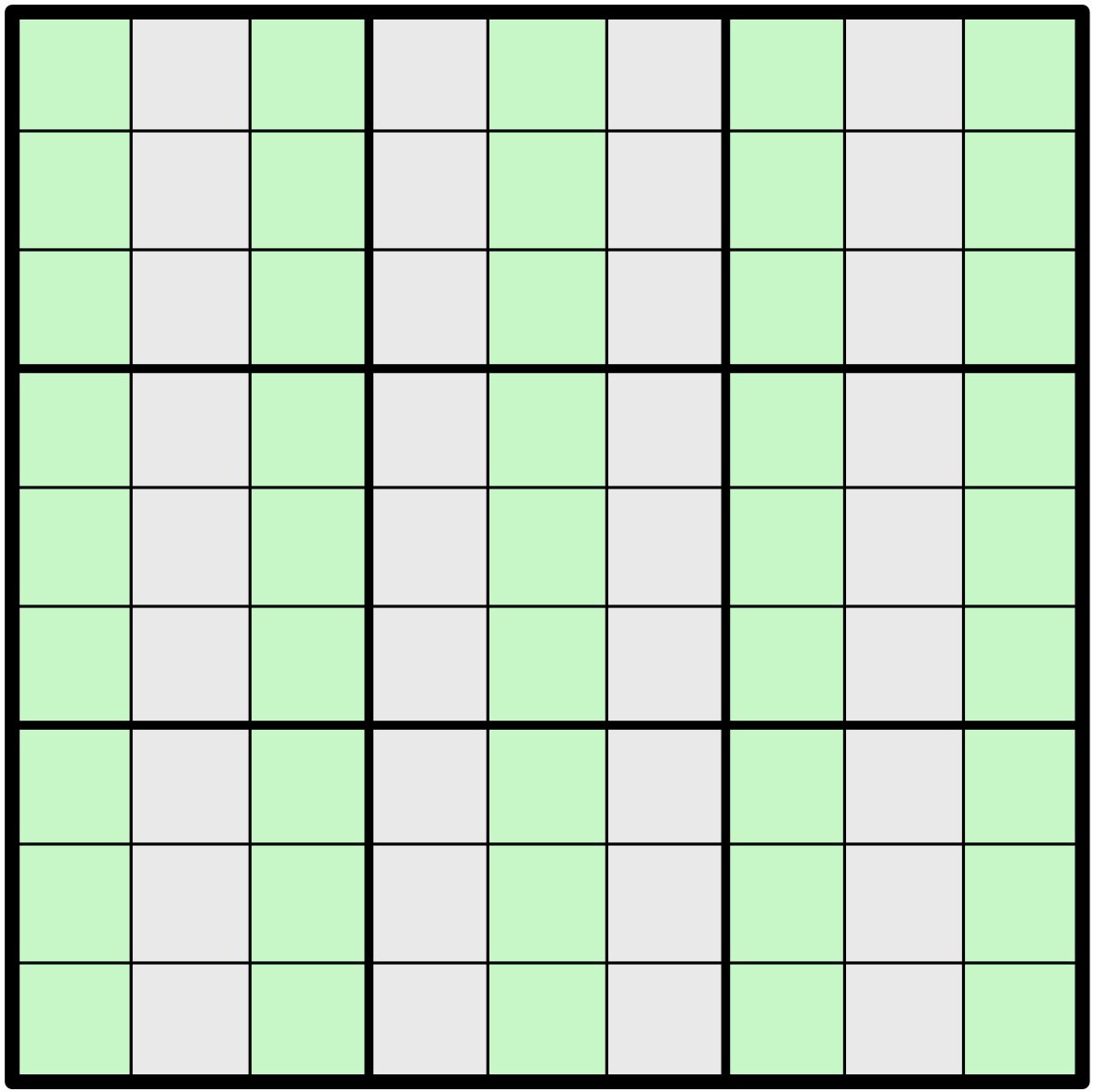


Sudoku



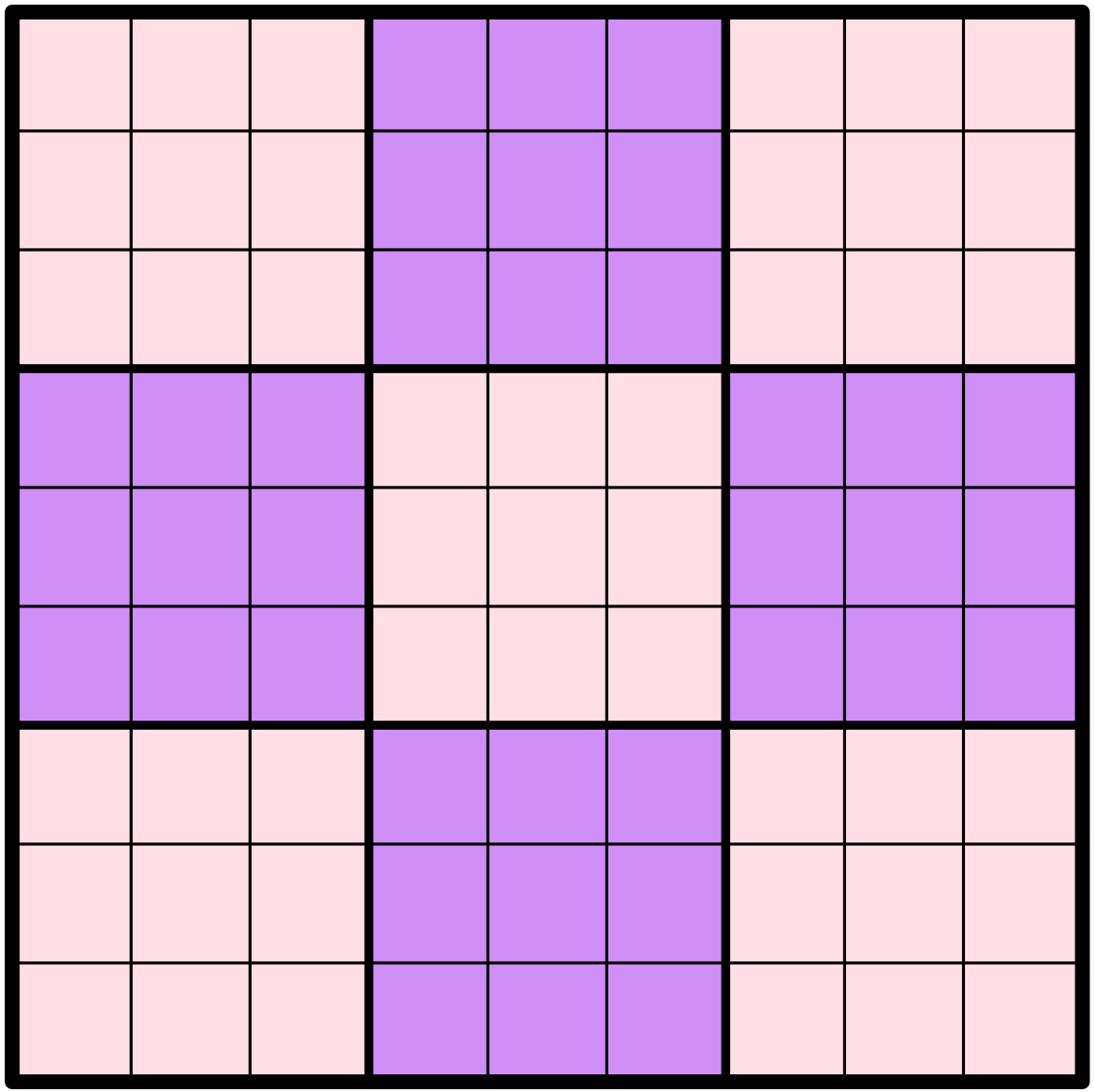
9 rows

Sudoku



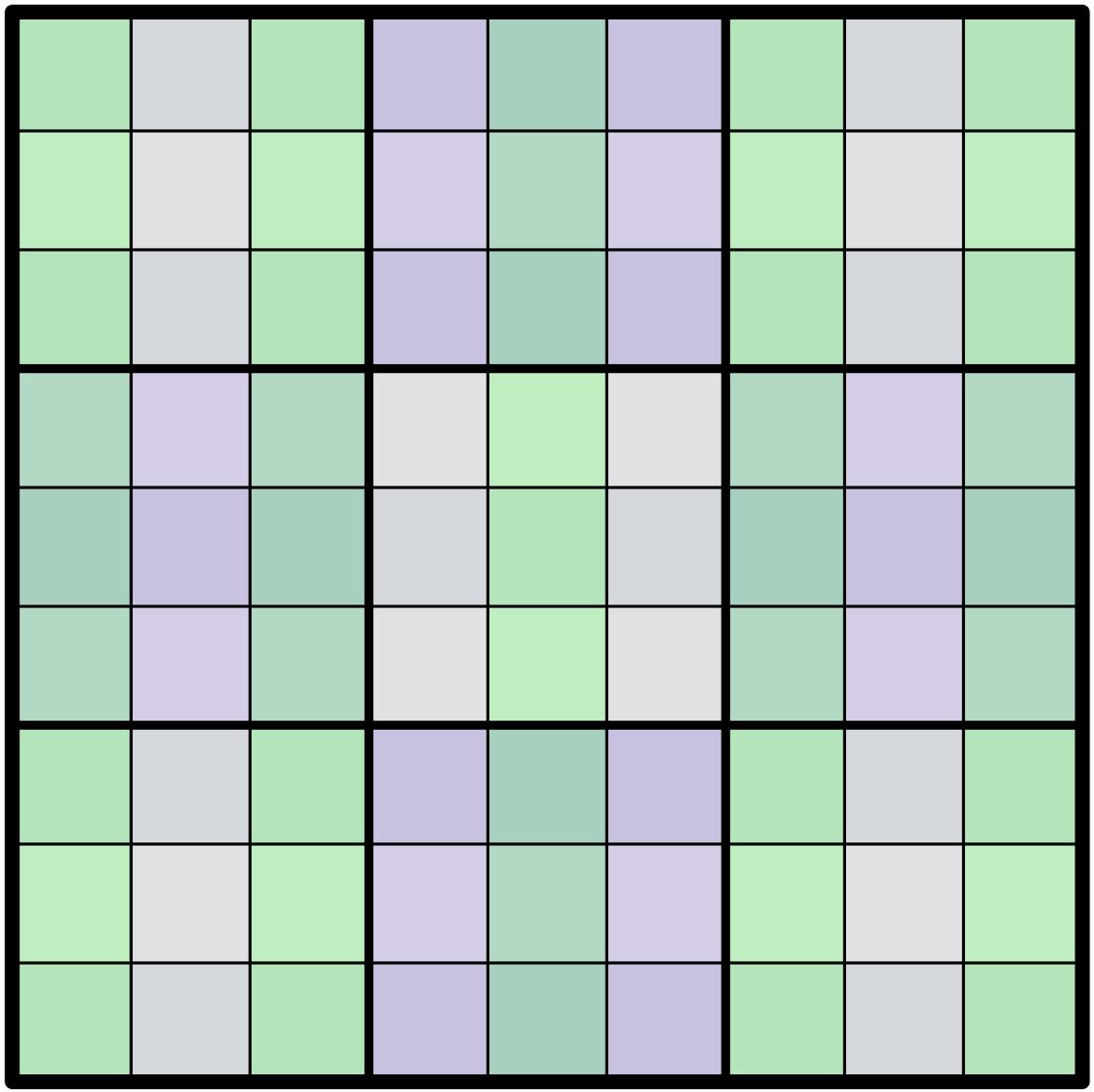
9 rows
9 columns

Sudoku



9 rows
9 columns
9 blocks

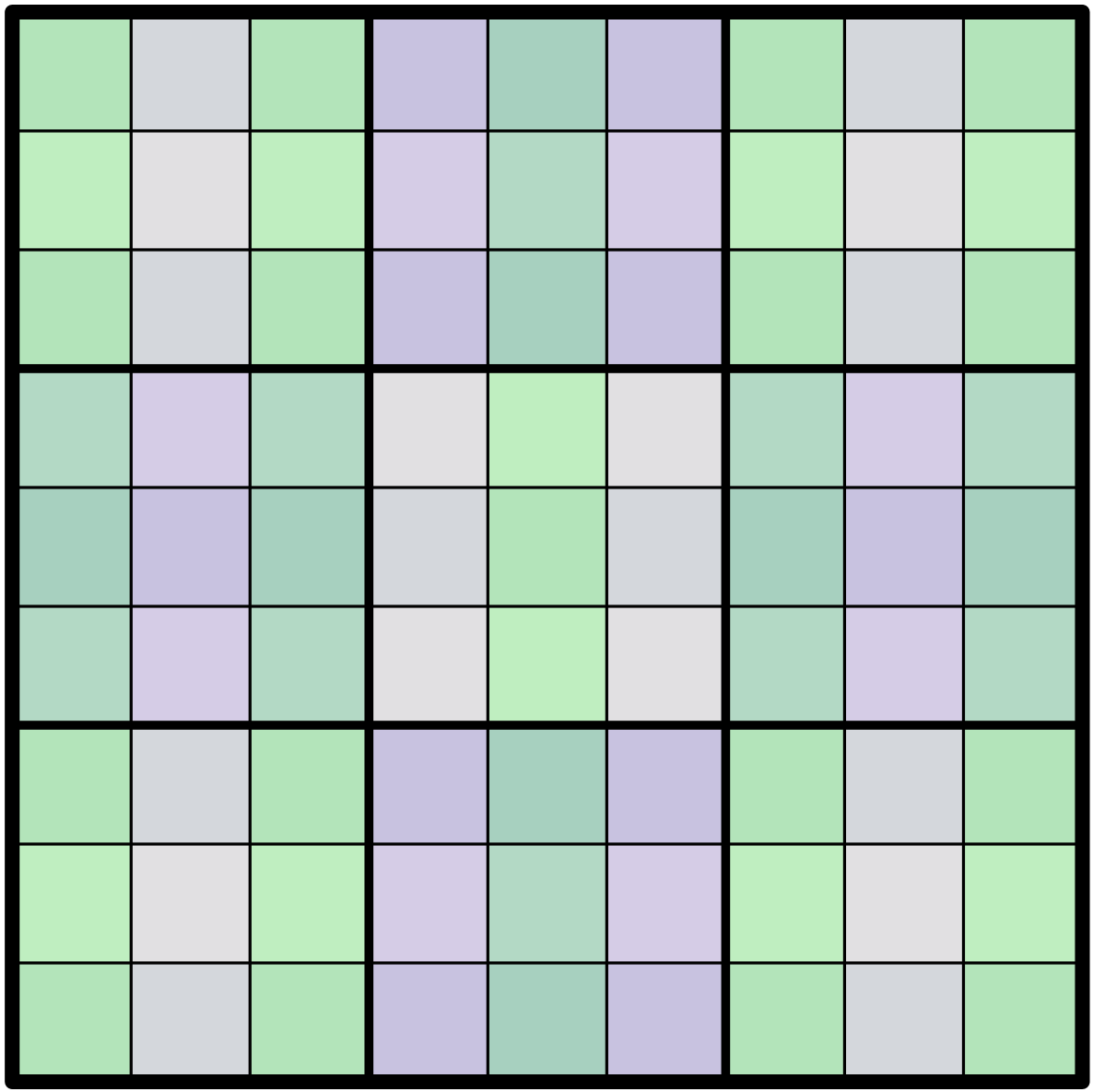
Sudoku



9 rows
9 columns
9 blocks

27 groups

Sudoku



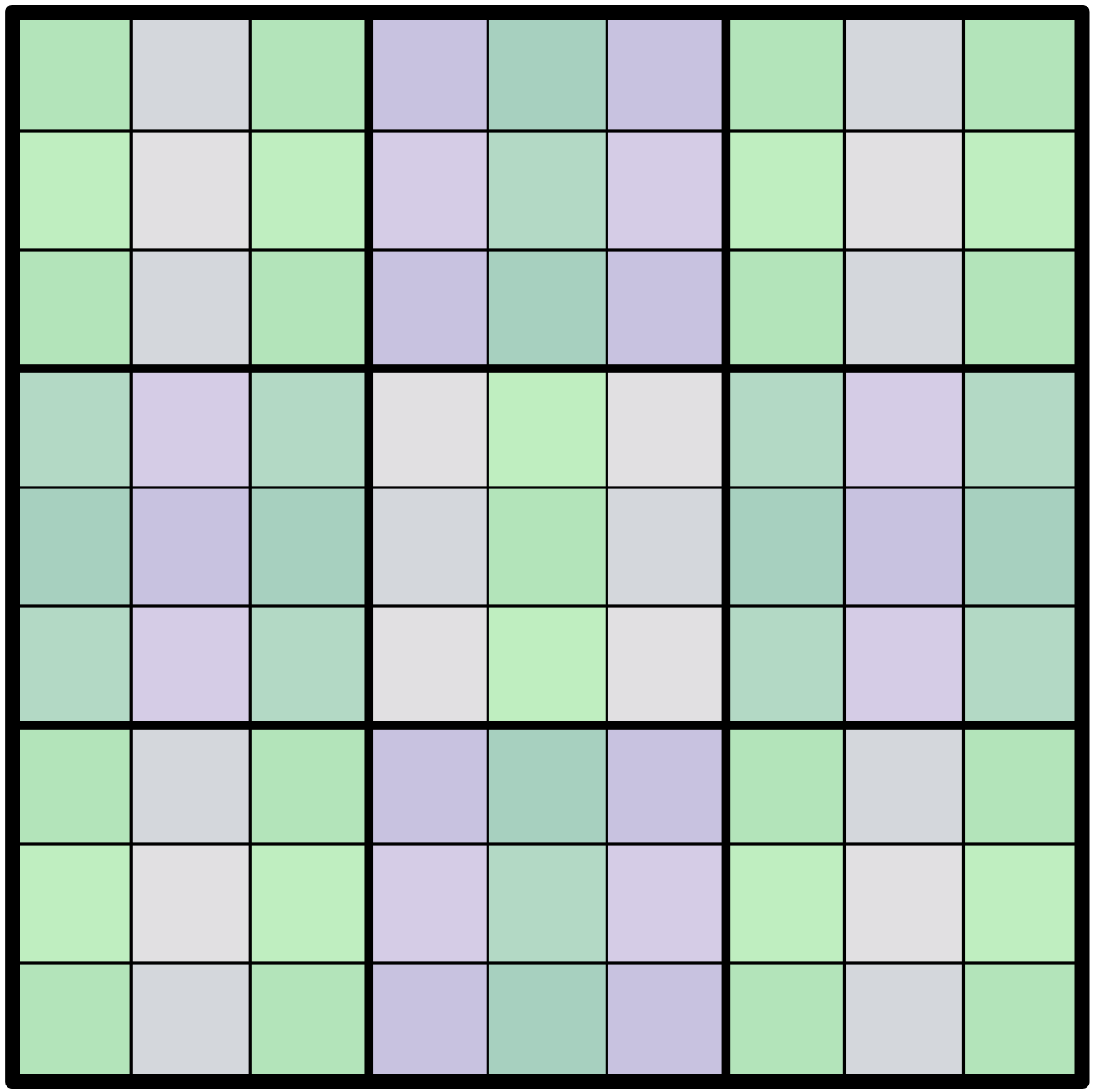
9 rows
9 columns
9 blocks

---

27 groups

each group
consists of
9 squares

Sudoku



9 rows
9 columns
9 blocks

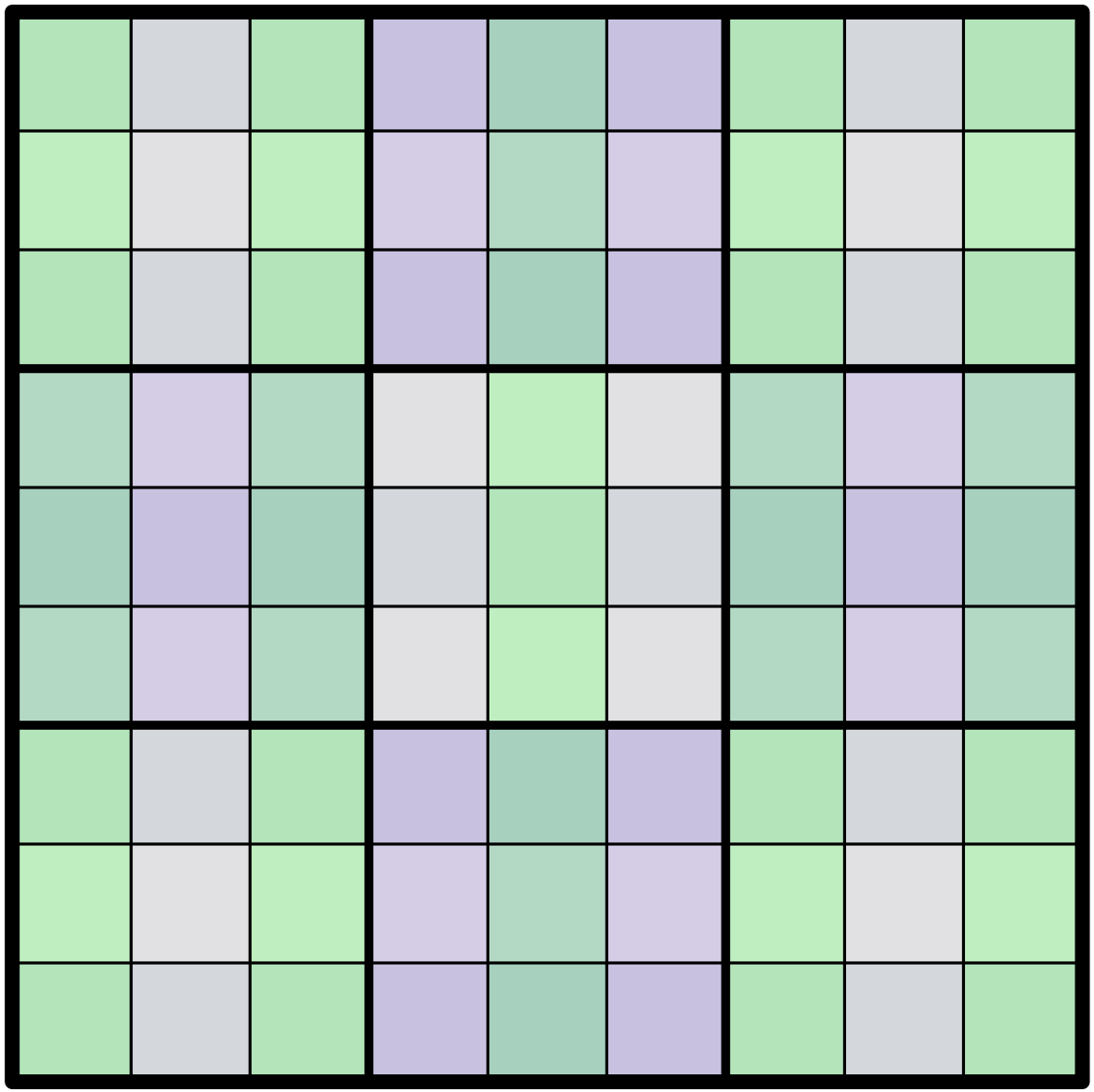
---

27 groups

each group consists of 9 squares

each square belongs to 3 groups

Sudoku



9 rows
9 columns
9 blocks

---

27 groups

each group
consists of
9 squares

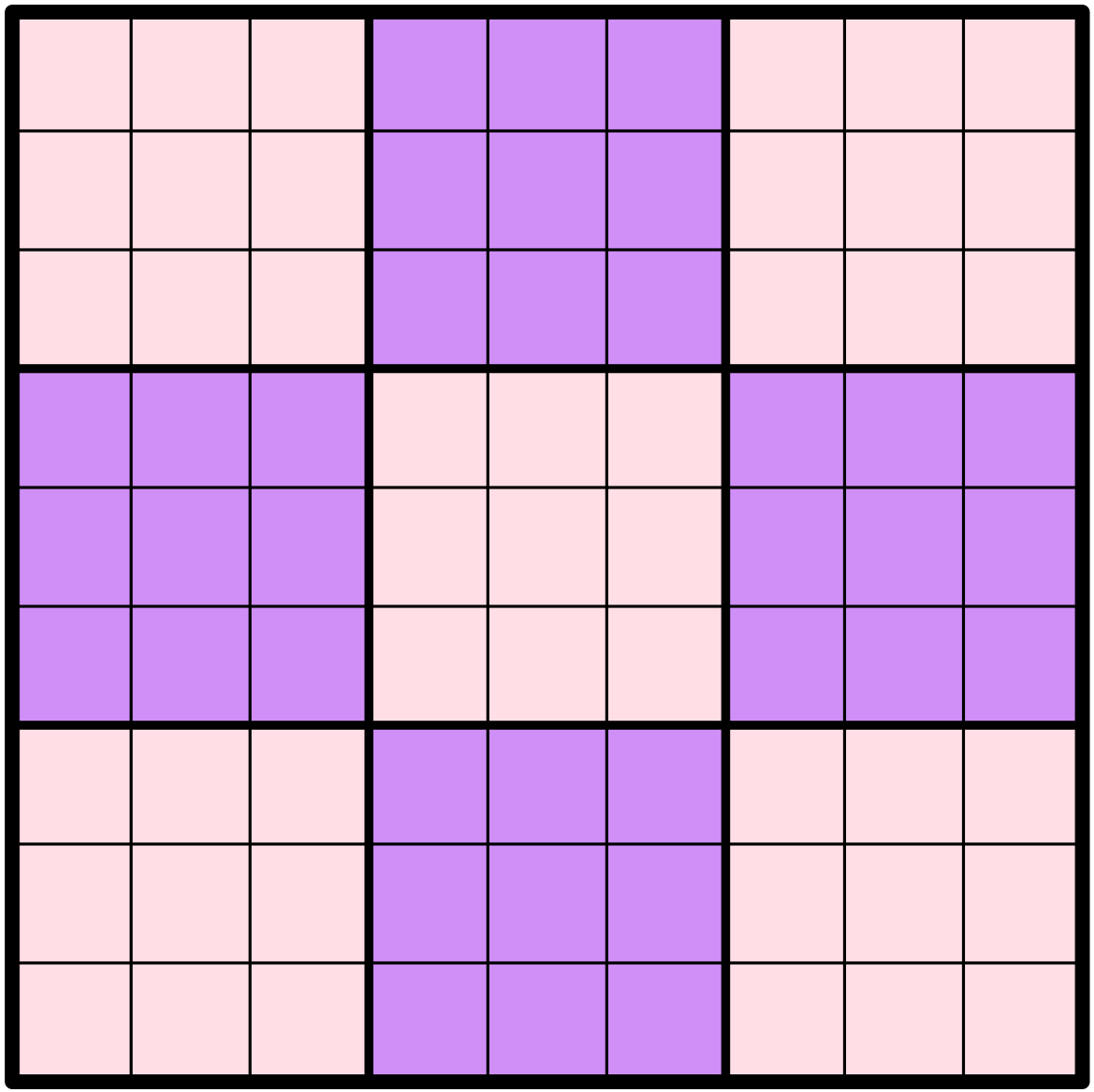
each square
belongs to
3 groups

---

total of
81 squares

Sudoku

Rule: each square gets a number from 1..9; each group must contain one square for each number 1..9



9 rows
9 columns
9 blocks

---

27 groups

each group consists of 9 squares

each square belongs to 3 groups

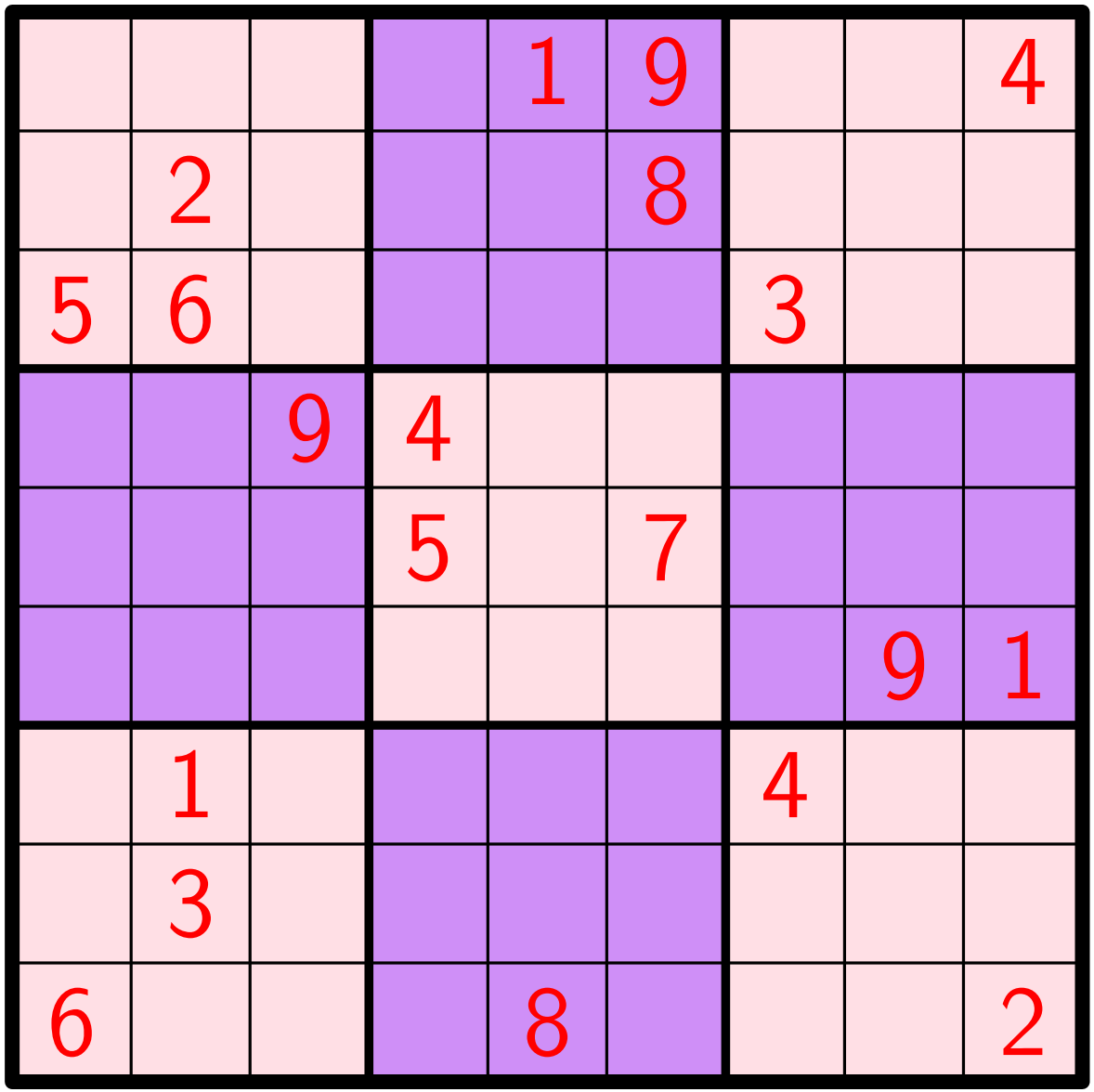
---

total of 81 squares

Sudoku

Rule: each square gets a number from 1..9; each group must contain one square for each number 1..9

We start with **givens**



9 rows
9 columns
9 blocks

27 groups

each group consists of 9 squares

each square belongs to 3 groups

total of 81 squares

Sudoku

Some facts:

- A valid sudoku must have a unique solution
- $\Rightarrow$ Givens must not lead to a contradiction
- At least 17 givens are needed (non trivial fact!)
- Even 77 givens (only 4 squares left) might not determine a unique solution
- Typical Sudokus have about 25-30 givens

Data Structure

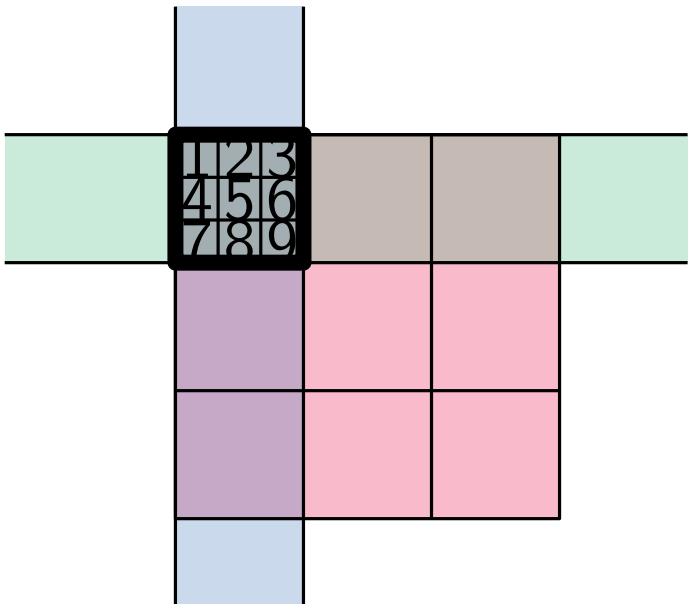
For each square sq (81×):

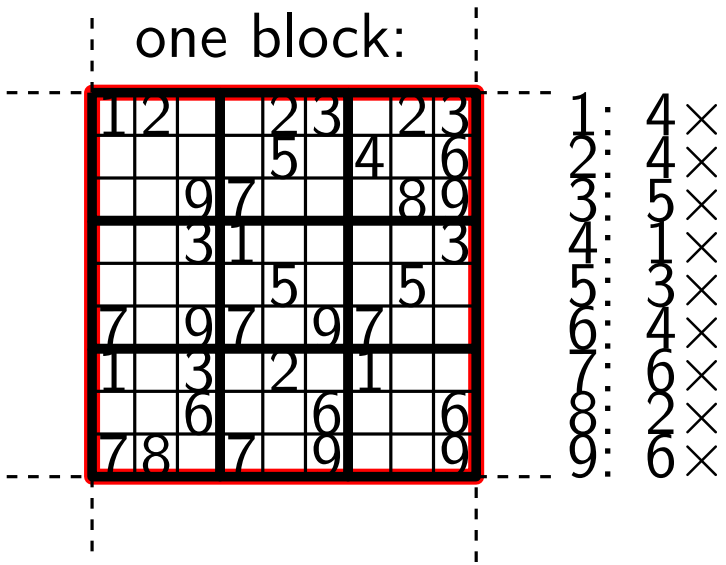
- still possible numbers for sq (initially 1,...,9)

For each group G (27×):

- For each number $i = 1, ..., 9$ a counter $c(G, i)$ of how many squares in G could still be i (initially 9)

Each square points to its 3 groups, each group to its 9 squares





Data Structure

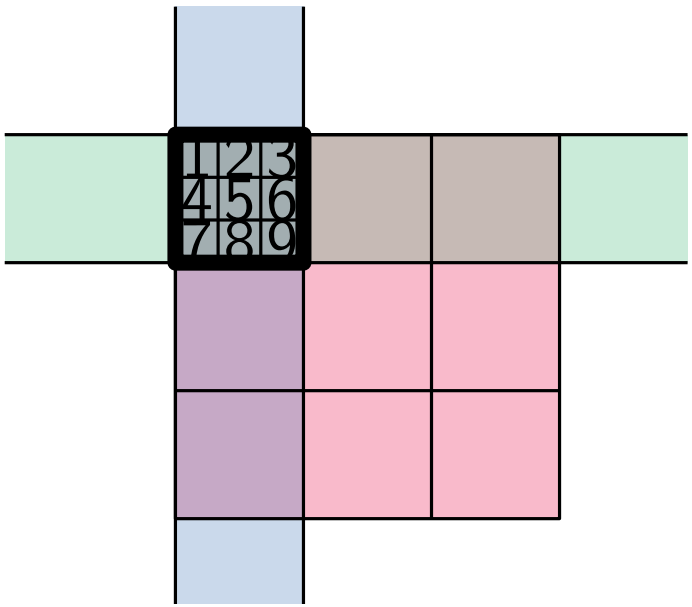
For each square sq (81×):

- still possible numbers for sq (initially 1,...,9)

For each group G (27×):

- For each number $i = 1, ..., 9$ a counter $c(G, i)$ of how many squares in G could still be i (initially 9)

Each square points to its 3 groups, each group to its 9 squares



Observation 1: If a square sq has only one possible number left, then we can fix it. Plus update of all groups of sq and the squares of these groups. And the groups of these squares.

Data Structure

For each square sq (81×):

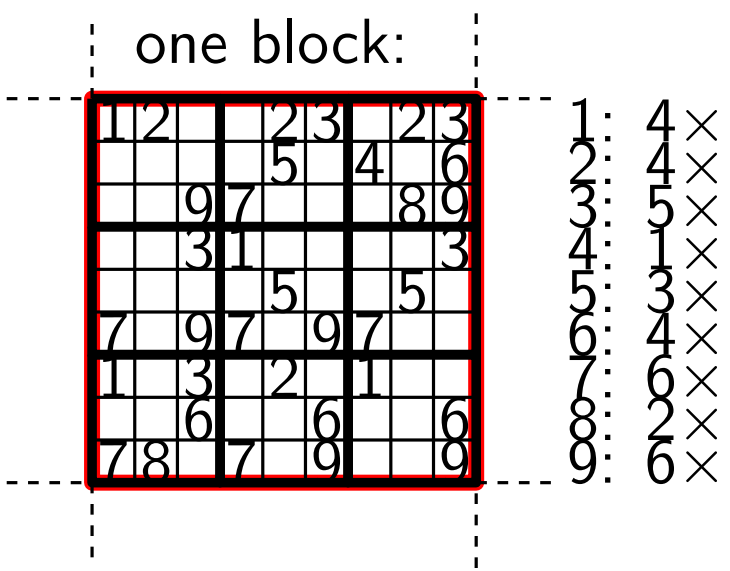
- still possible numbers for sq (initially 1,...,9)

For each group G (27×):

- For each number $i = 1, ..., 9$ a counter $c(G, i)$ of how many squares in G could still be i (initially 9)

Each square points to its 3 groups, each group to its 9 squares

Observation 2: If a counter $c(G, i)$ is one, then we can fix this number in a square sq in G. Plus update of all groups of sq and the squares of these groups.



Solving Level 1 Sudokus: Pseudocode

PROCEDURE FIX-SQUARE (sq, nr)

Set square sq to nr
For all three groups G which contain sq
    For all $nr' \neq nr$ if nr' still possible for sq DO
        reduce counter $c(G, nr')$
    For all squares sq' of G, $sq' \neq sq$ DO
        remove nr from sq'
        FOR each group $G' \neq G$ which contains sq' DO
            reduce counter $c(G', nr)$
For all sq', not fixed and only one possible number nr' DO
    fix-square(sq', nr')
For all $c(G, nr)$ which have been reduced to 1 DO
    search for square sq' of G which still allows nr
    fix-square(sq', nr)

Solving Level 1 Sudokus: Pseudocode

- Call FIX-SQUARE() for all given numbers of a sudoku
- If all 81 squares get fixed, the Sudoku is solved

Solving Level 1 Sudokus: Pseudocode

- Call FIX-SQUARE() for all given numbers of a sudoku
- If all 81 squares get fixed, the Sudoku is solved

- For all simple (Level 1) Sudokus FIX-SQUARE() will provide the solution

Solving Level 1 Sudokus: Pseudocode

- Call FIX-SQUARE() for all given numbers of a sudoku
- If all 81 squares get fixed, the Sudoku is solved

- For all simple (Level 1) Sudokus FIX-SQUARE() will provide the solution

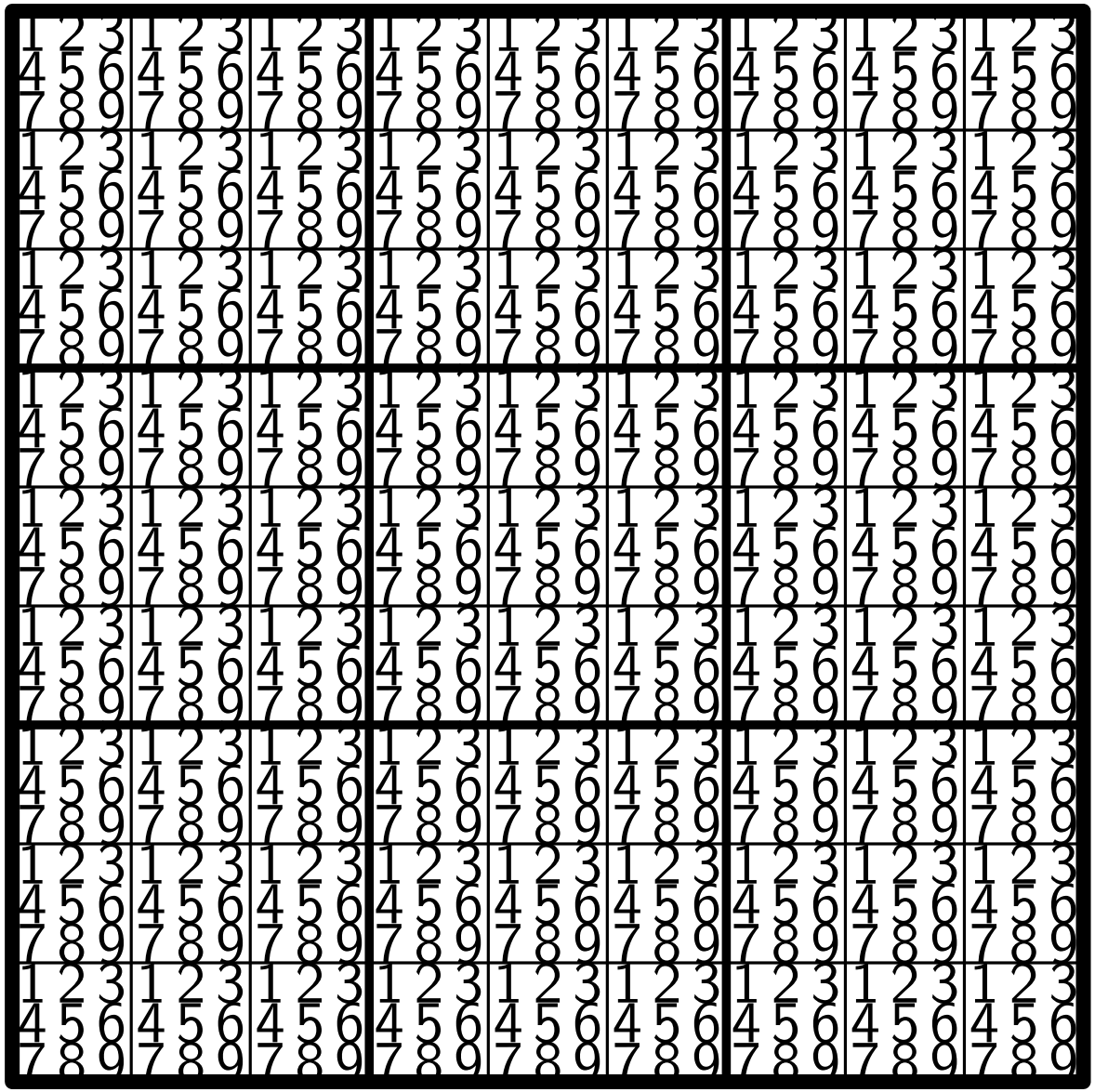
- Add check if a counter $c(G, nr)$ gets zero or if a square has no valid number left $\Rightarrow$ return('No valid solution')
- Thus we can call FIX-SQUARE() in a procedure with backtracking
- This will solve any valid sudoku

Sudoku 1

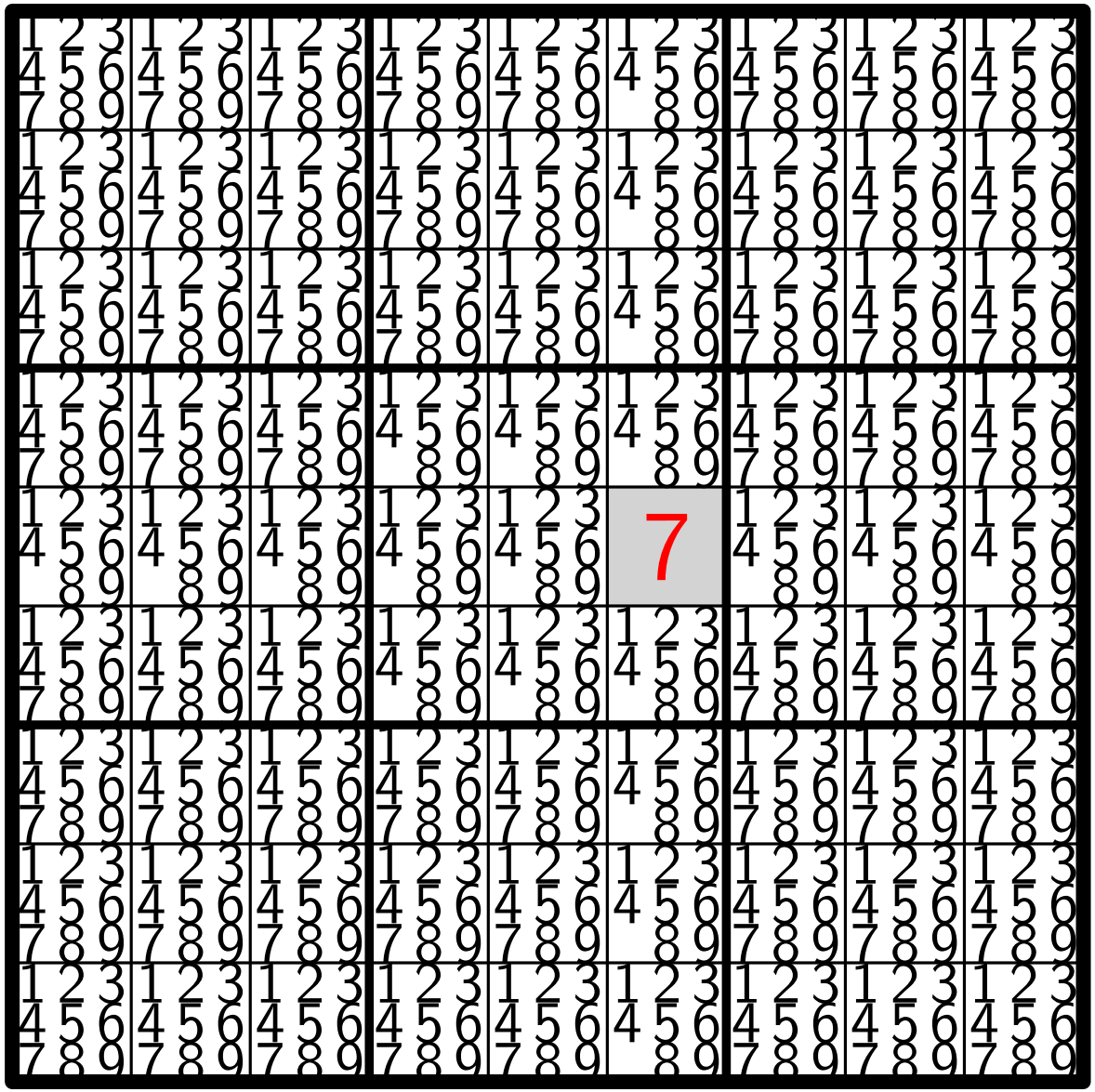
20 givens:

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

Sudoku 1

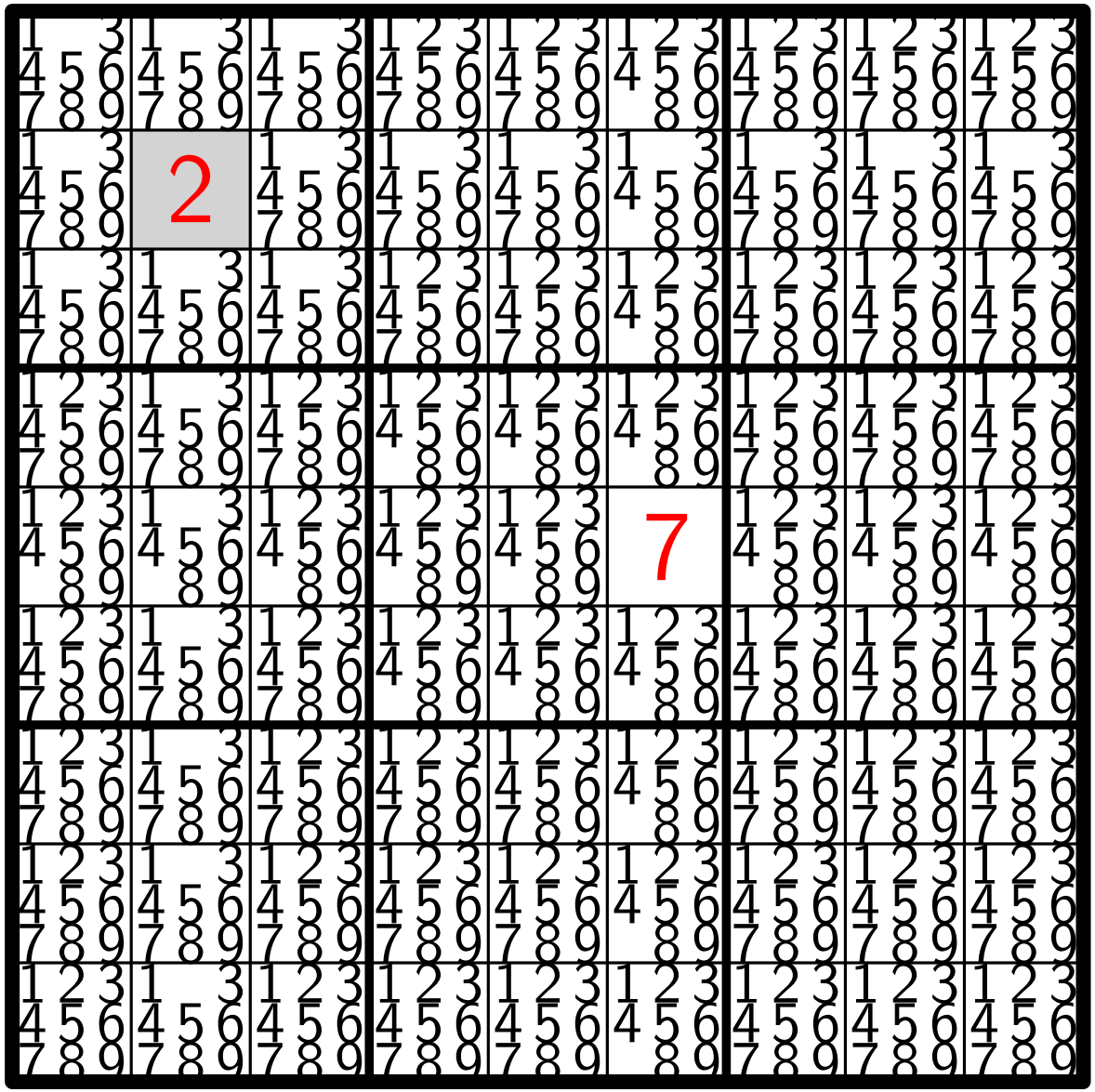


Sudoku 1



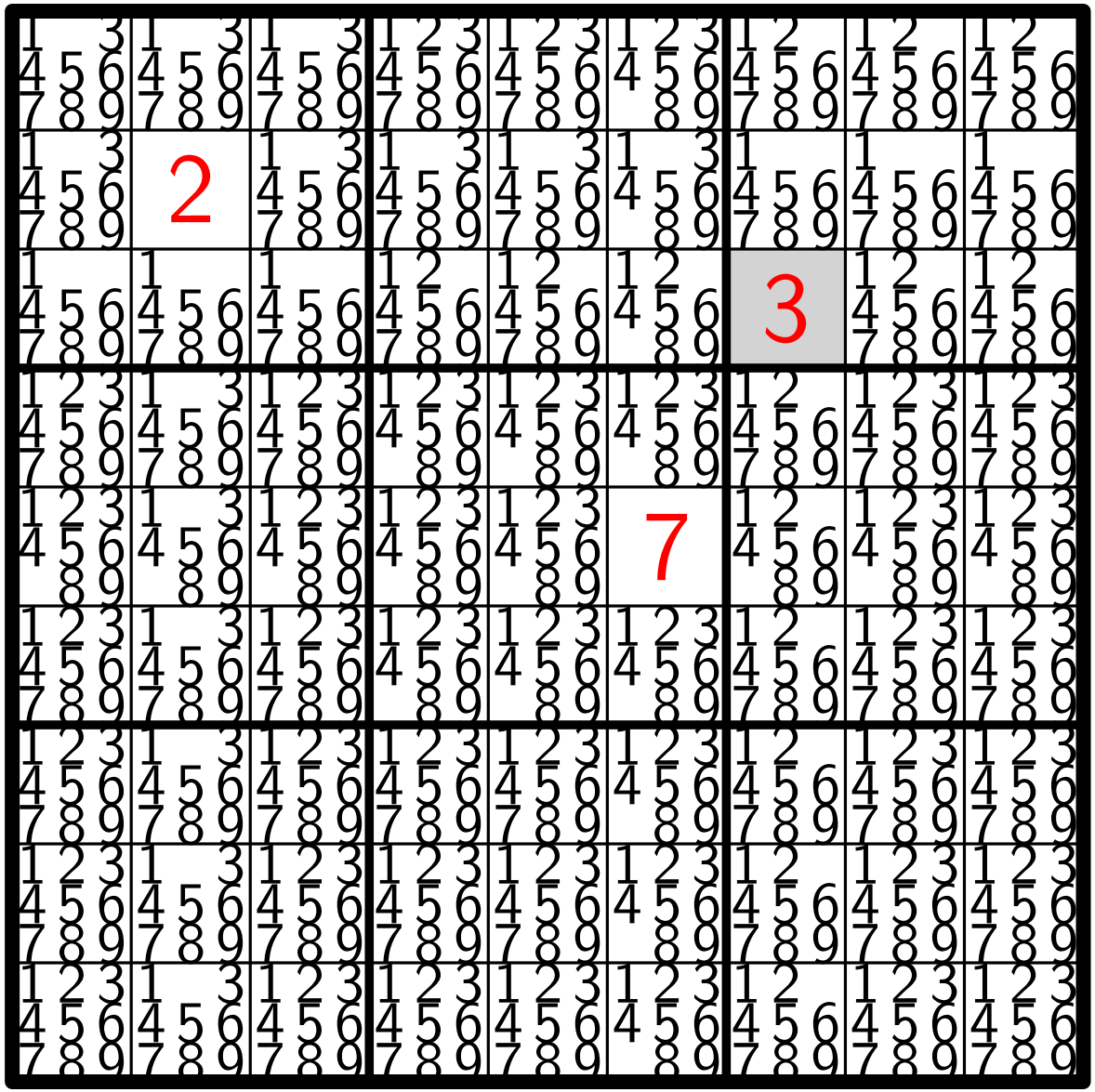
1

Sudoku 1



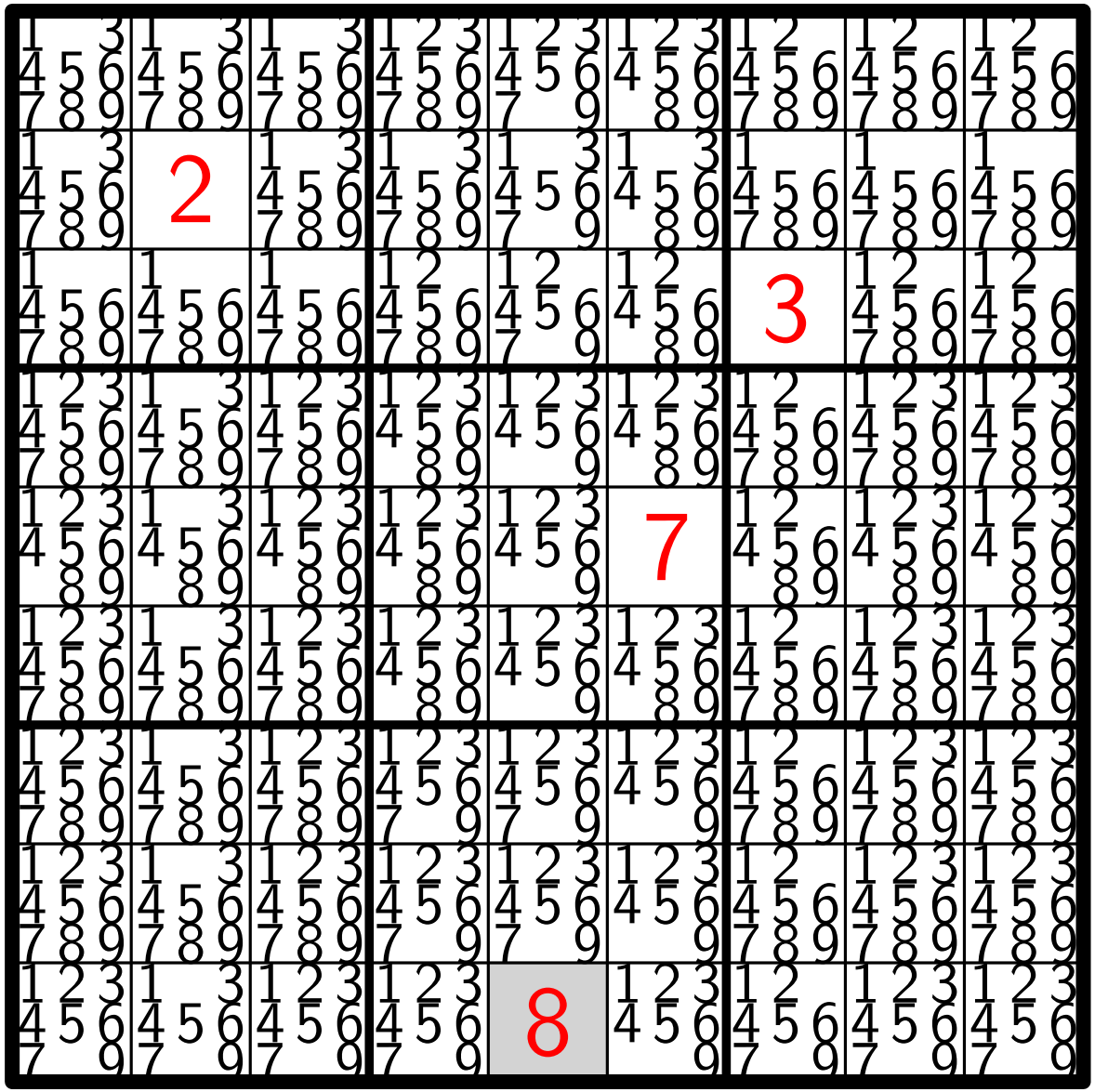
2

Sudoku 1



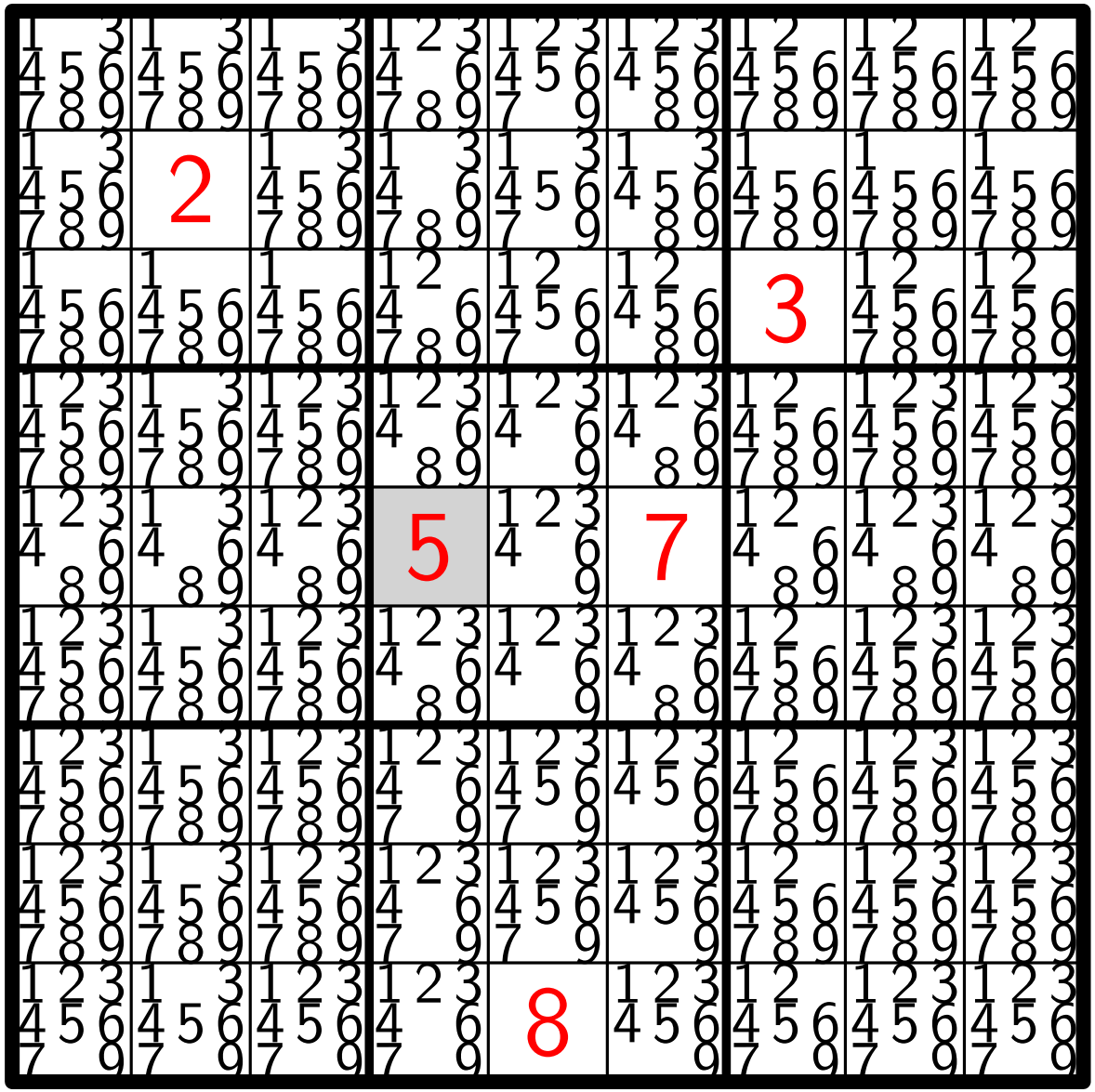
3

Sudoku 1



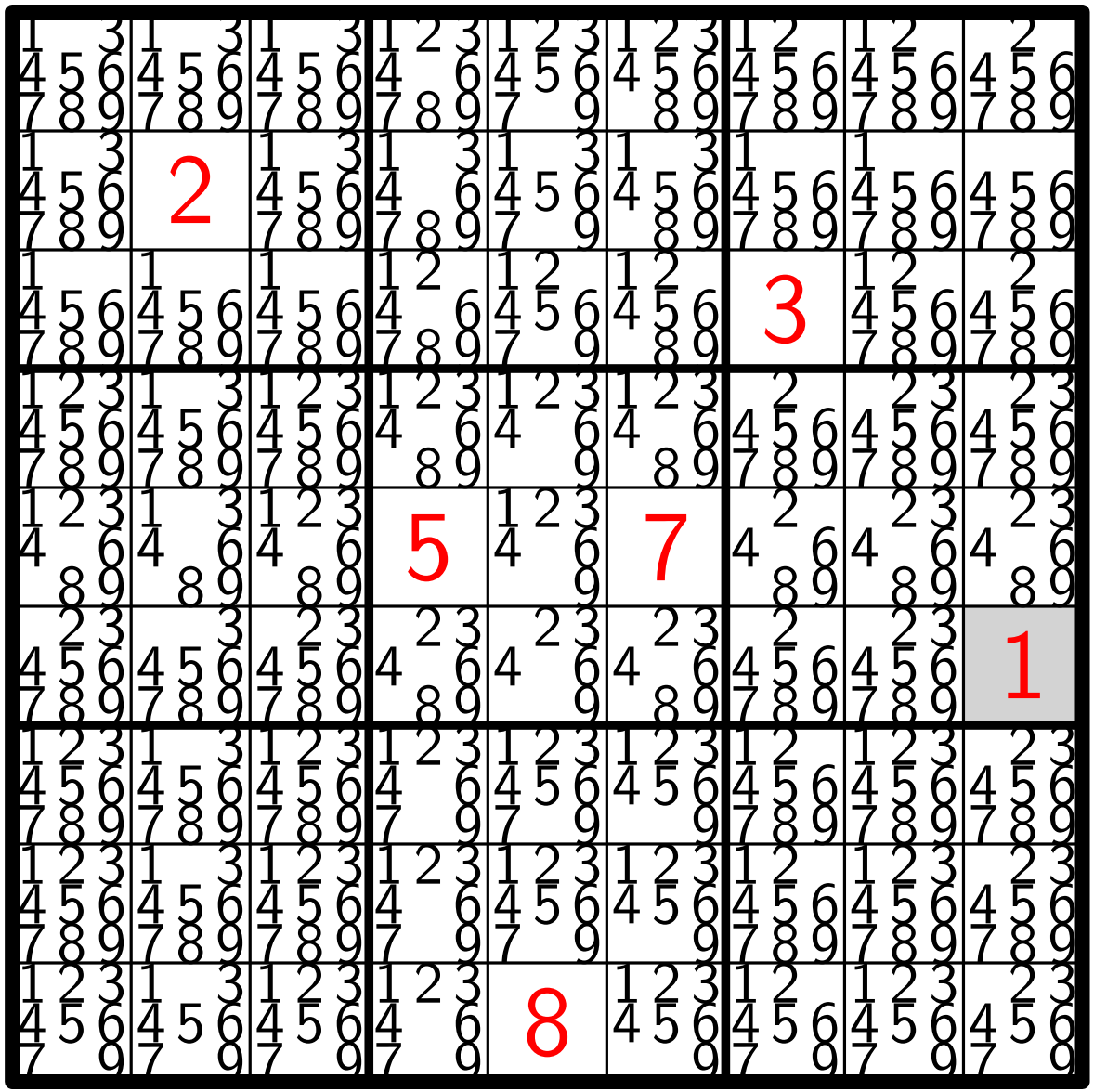
4

Sudoku 1



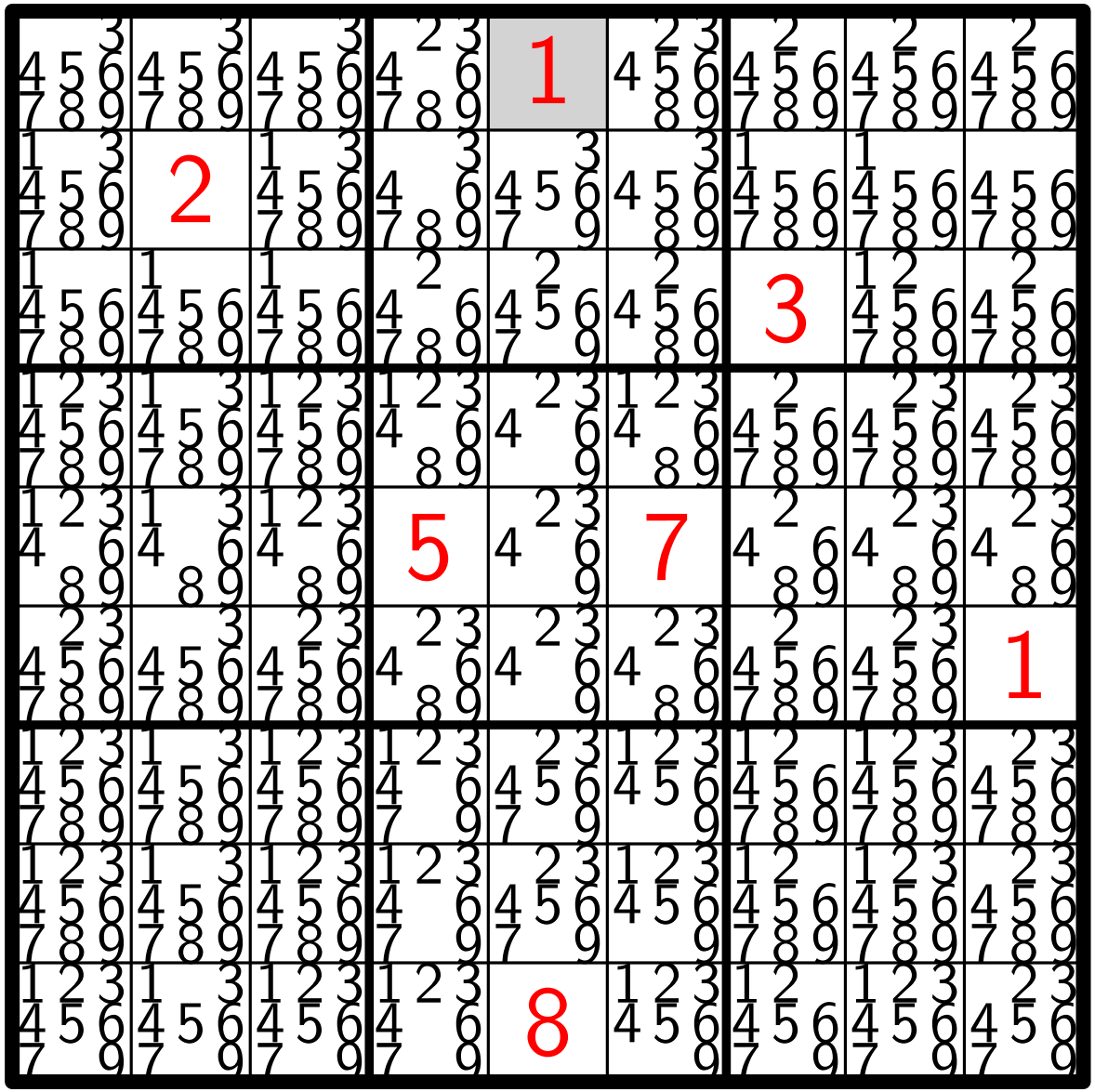
5

Sudoku 1



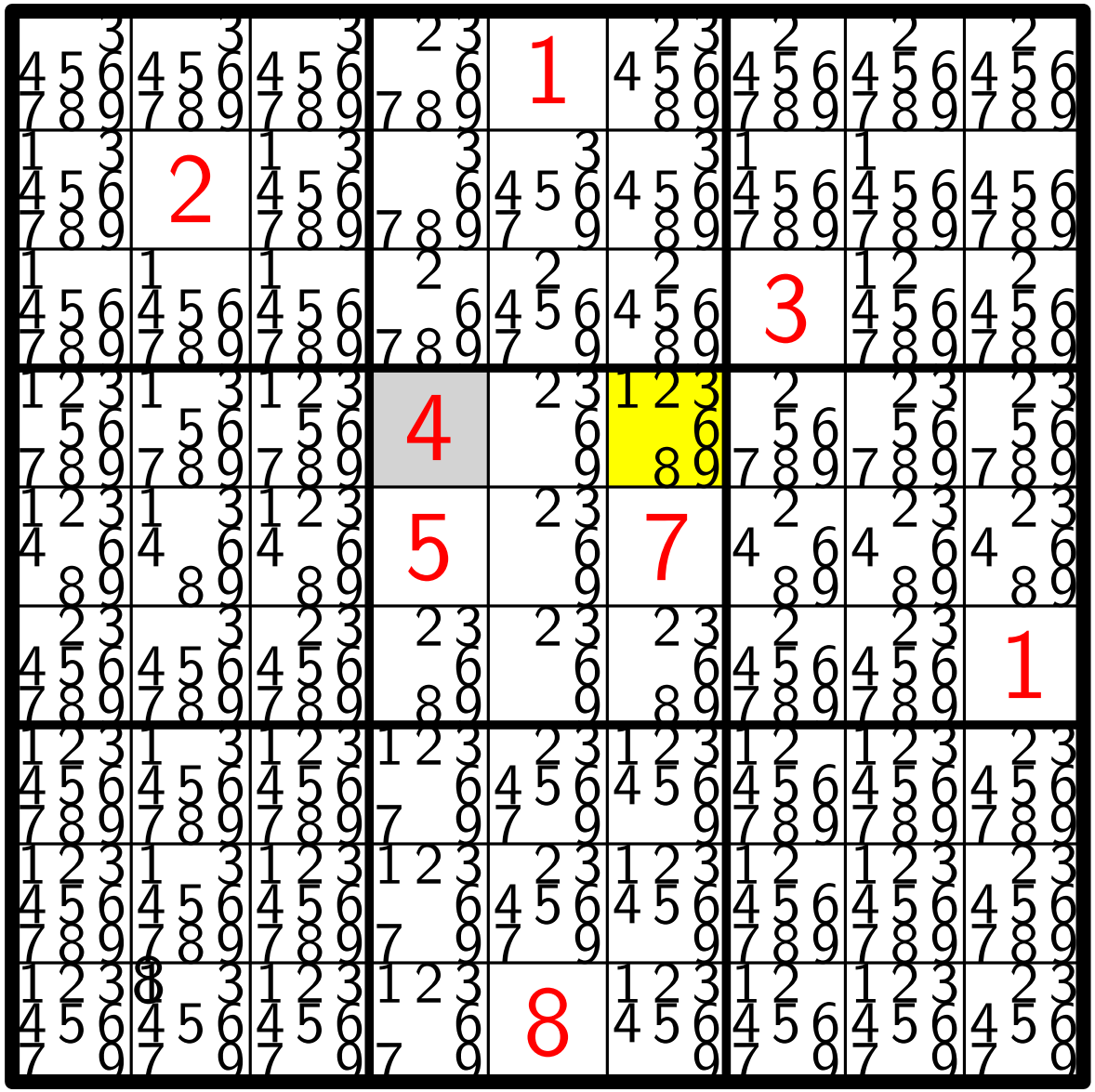
6

Sudoku 1



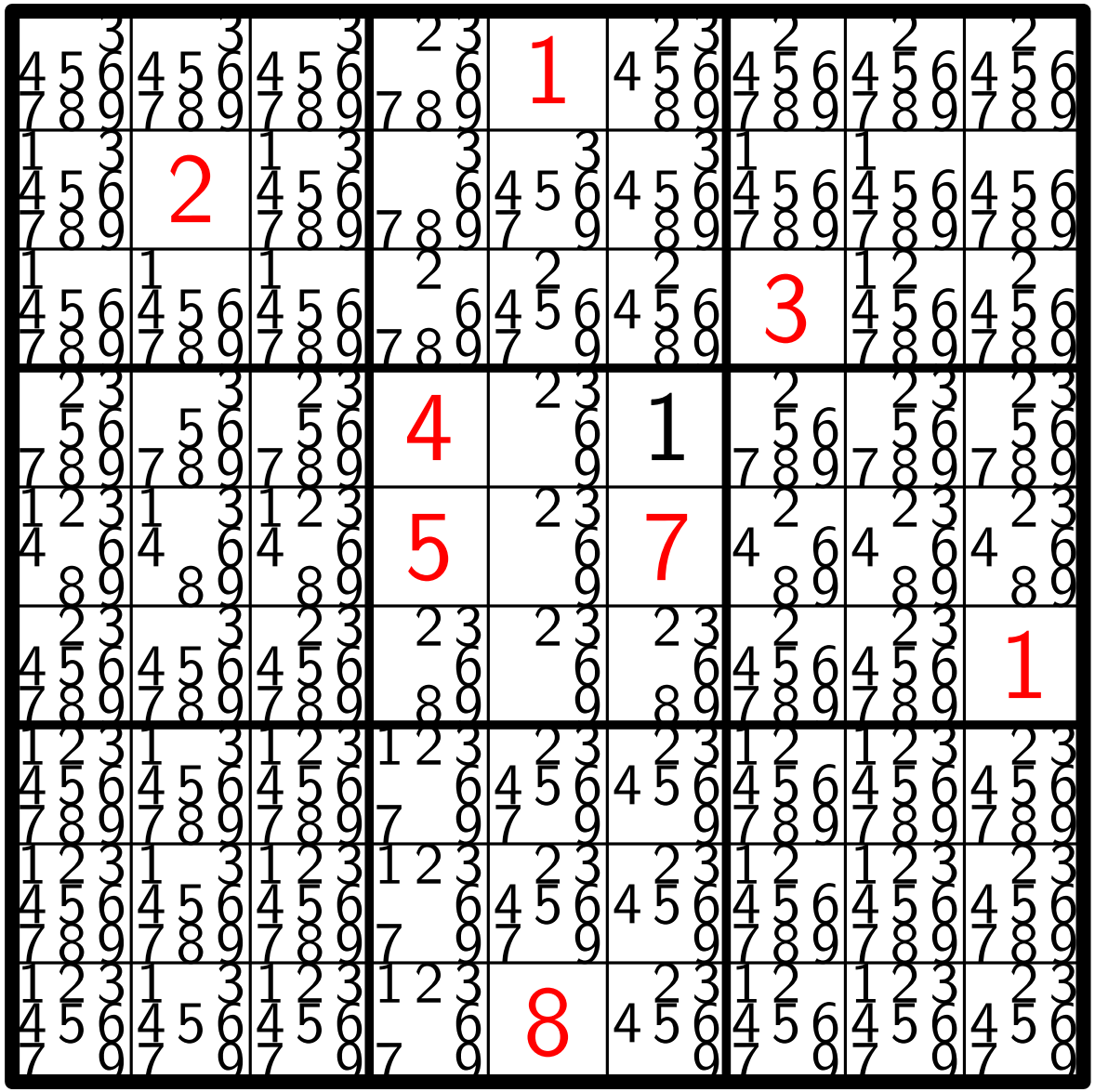
7

Sudoku 1

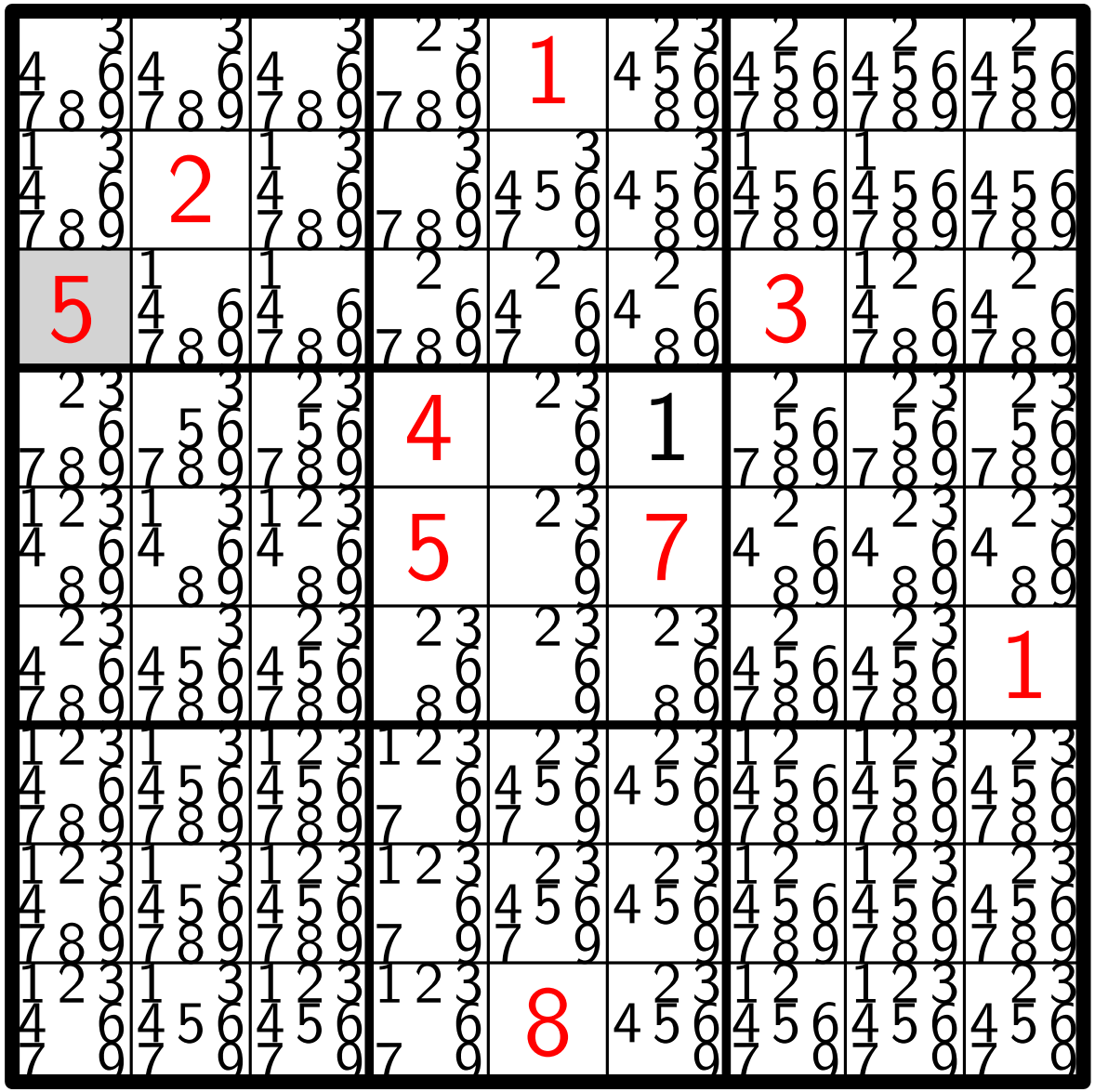


8

Sudoku 1

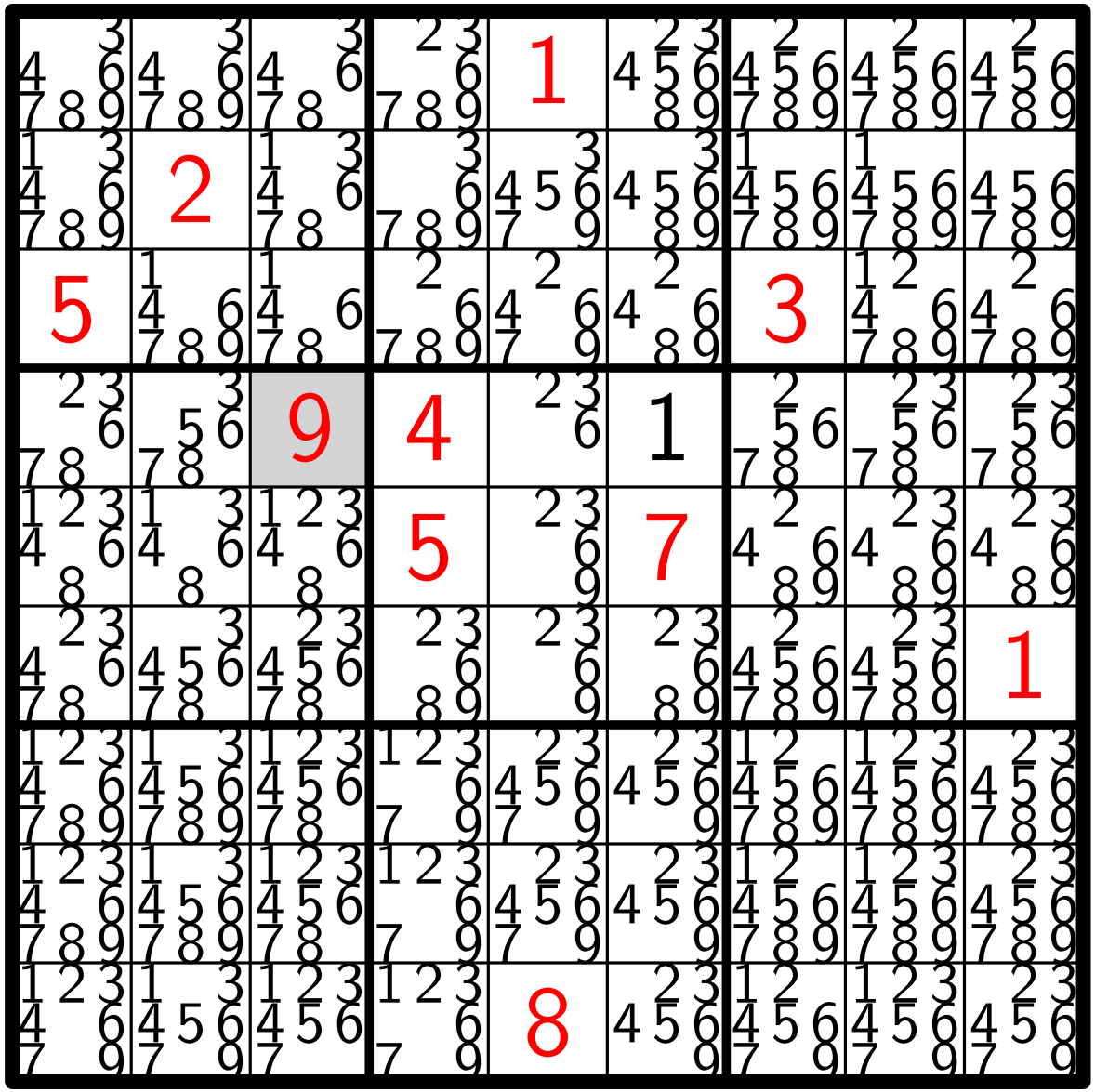


Sudoku 1



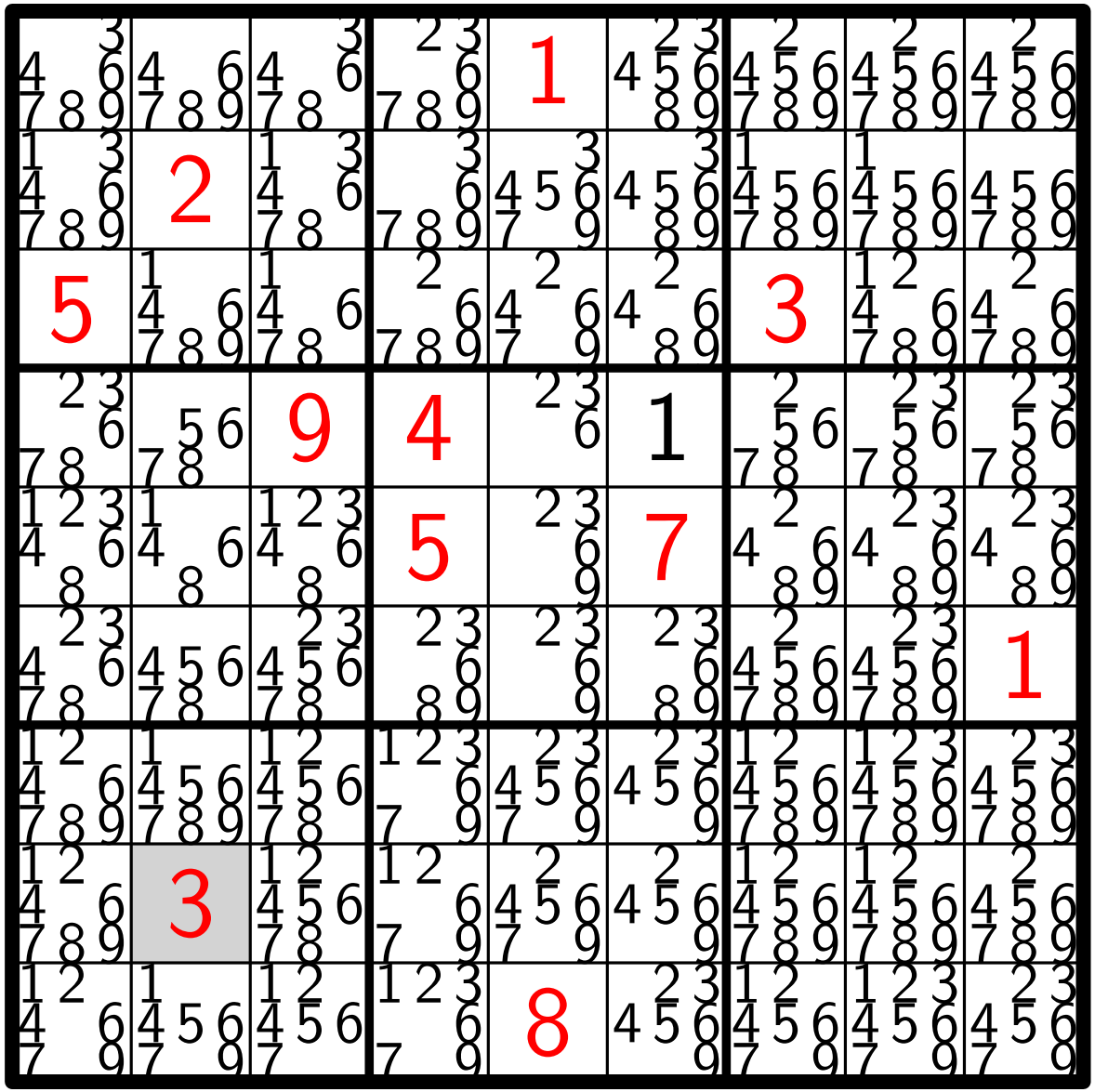
9

Sudoku 1



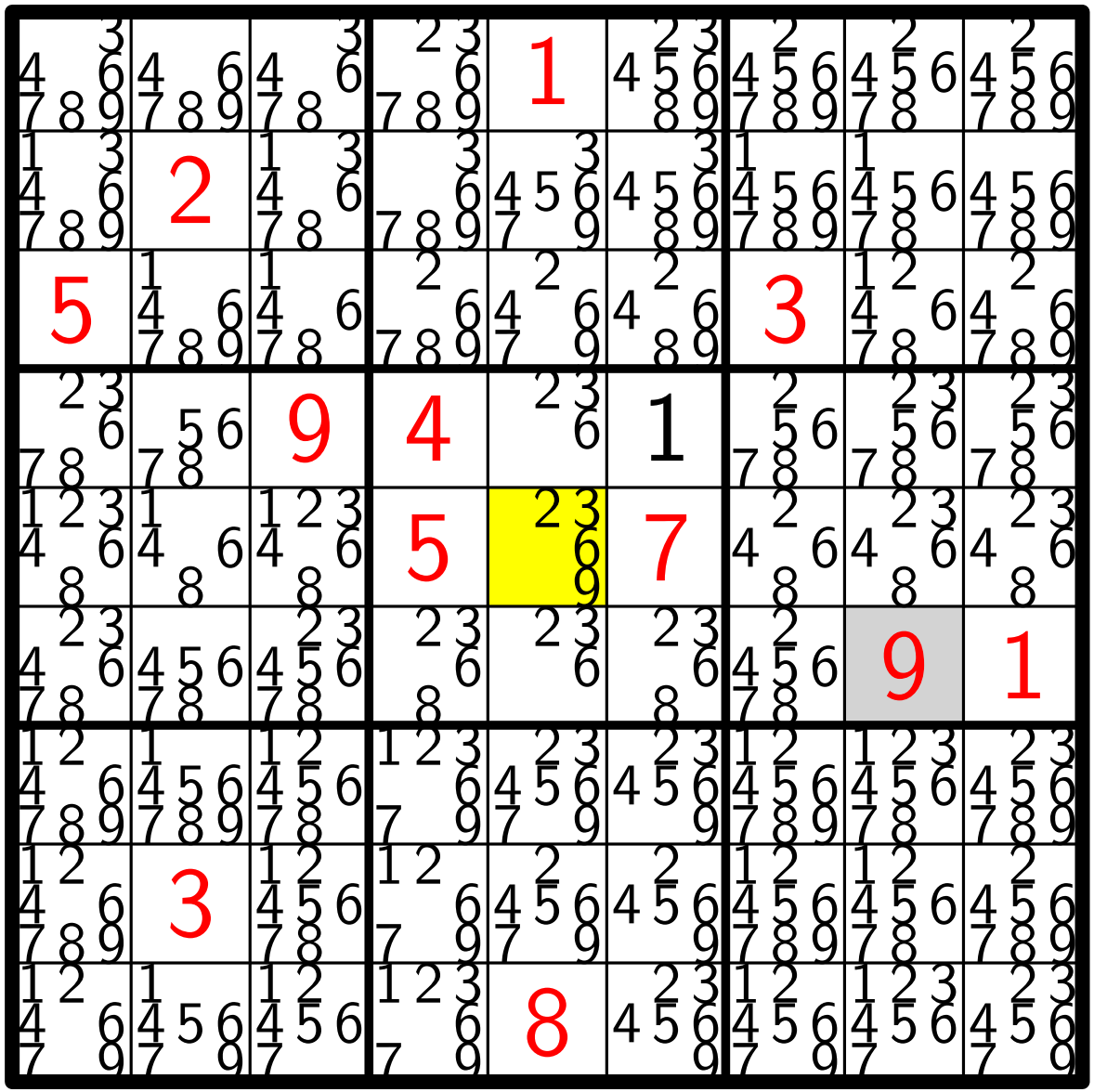
10

Sudoku 1



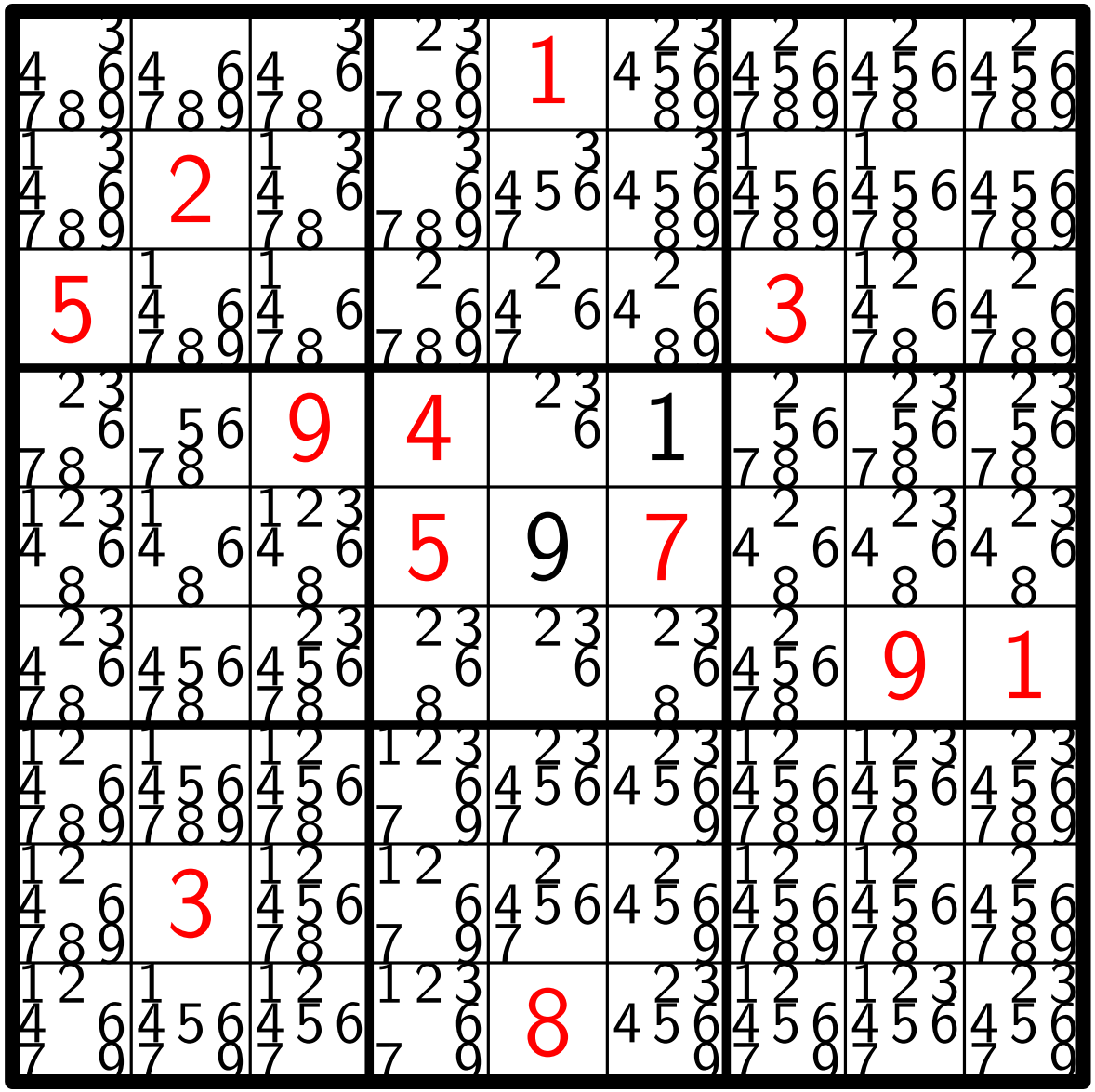
11

Sudoku 1



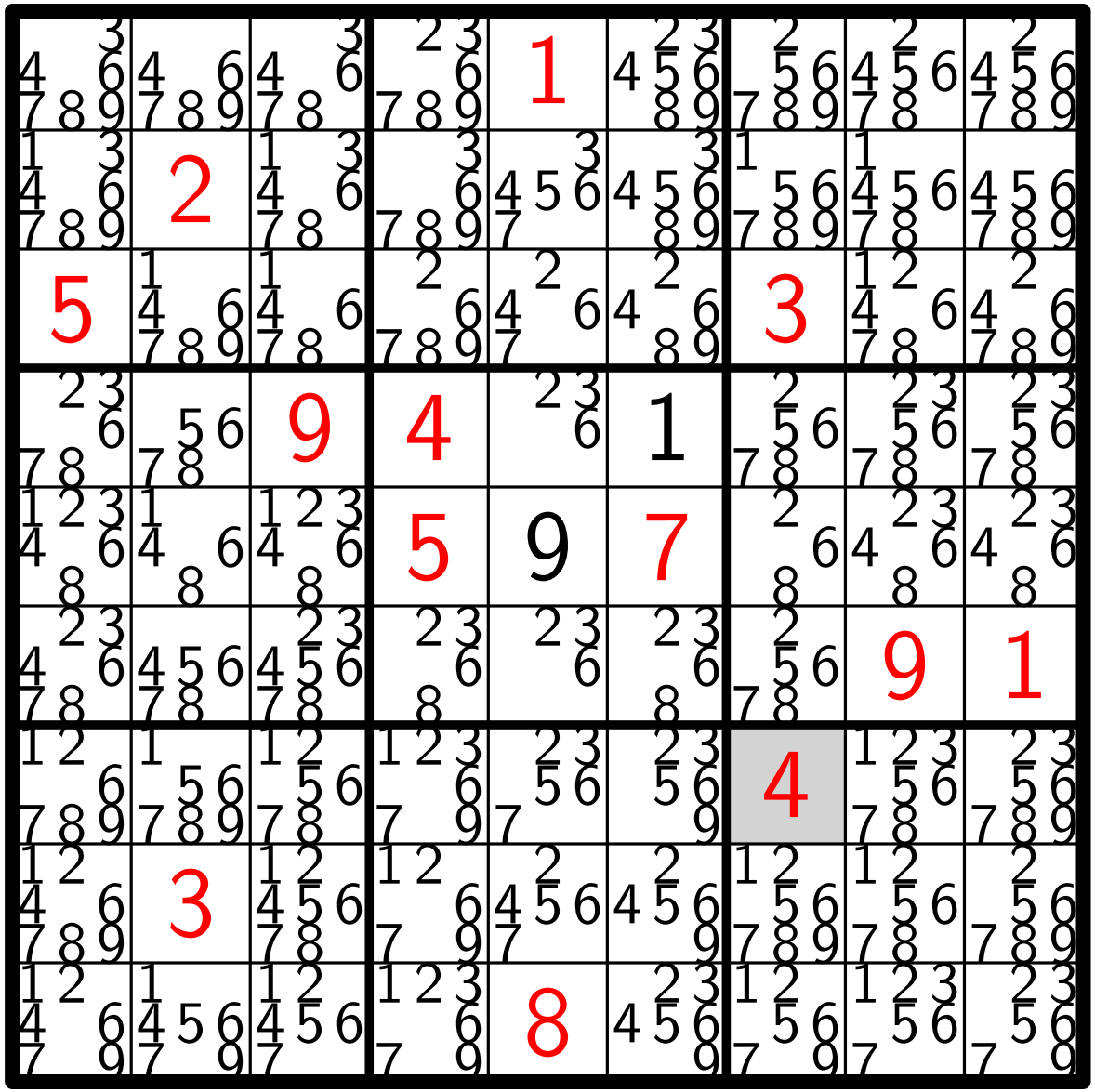
12

Sudoku 1



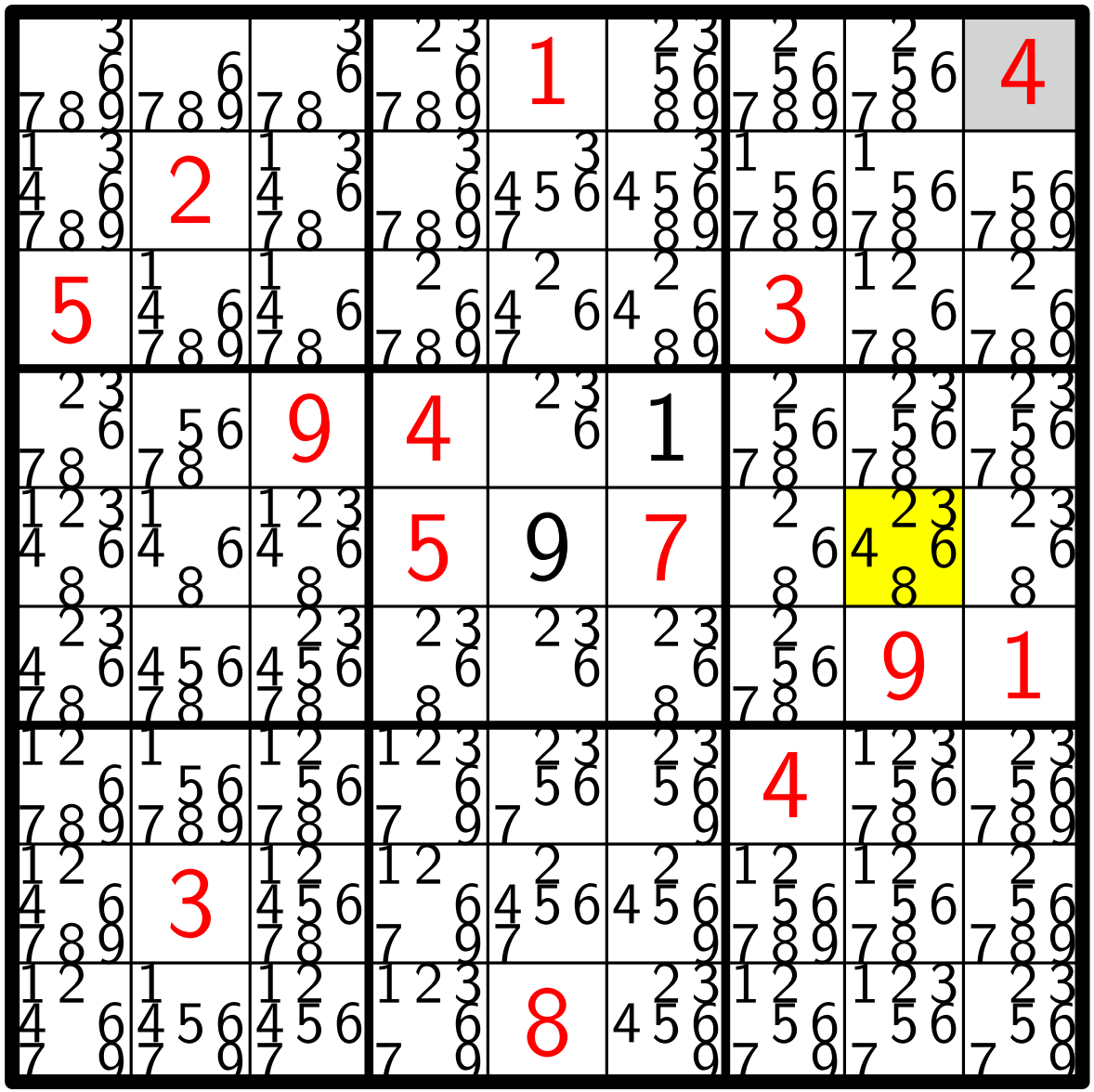
12

Sudoku 1



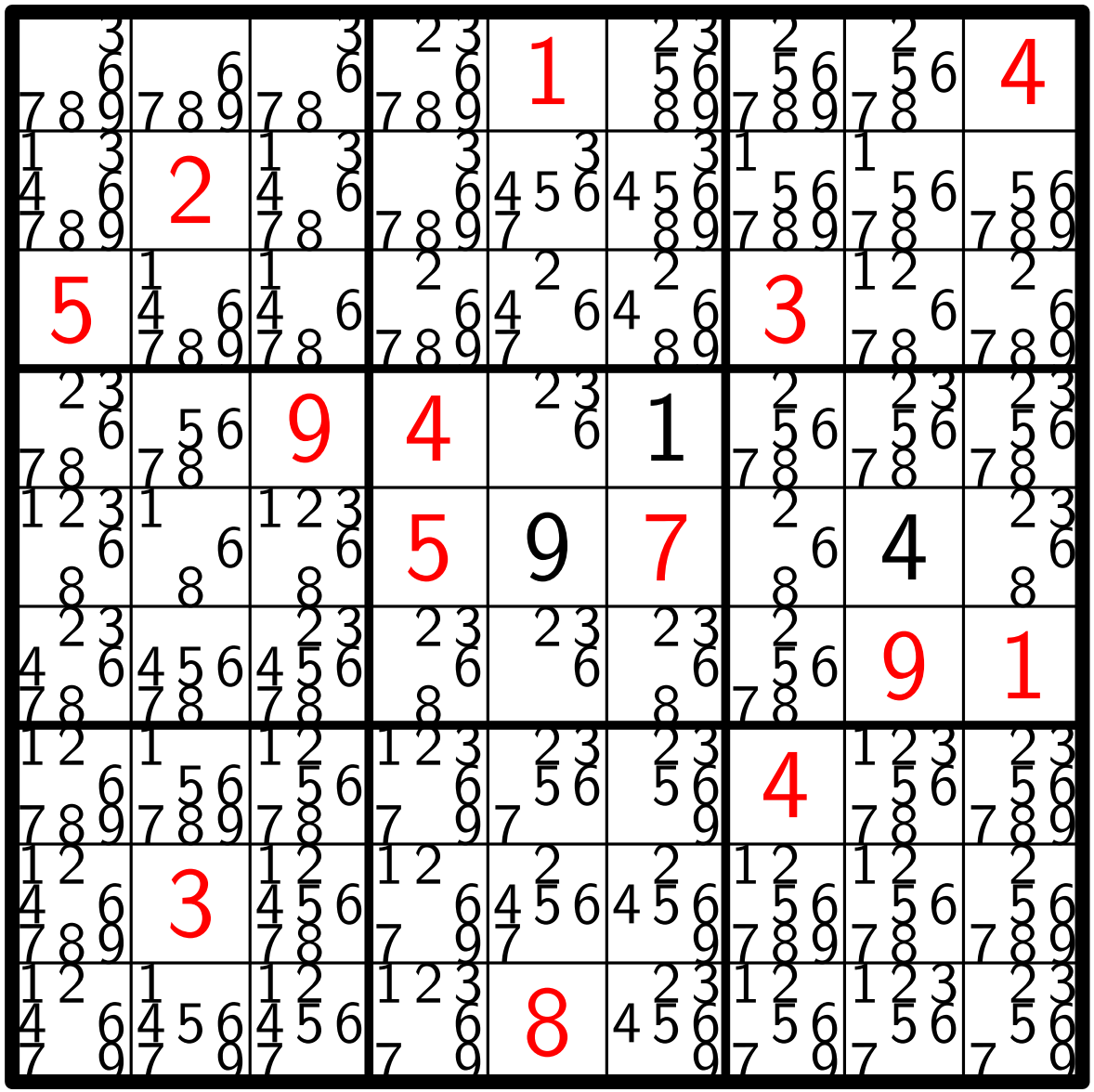
13

Sudoku 1



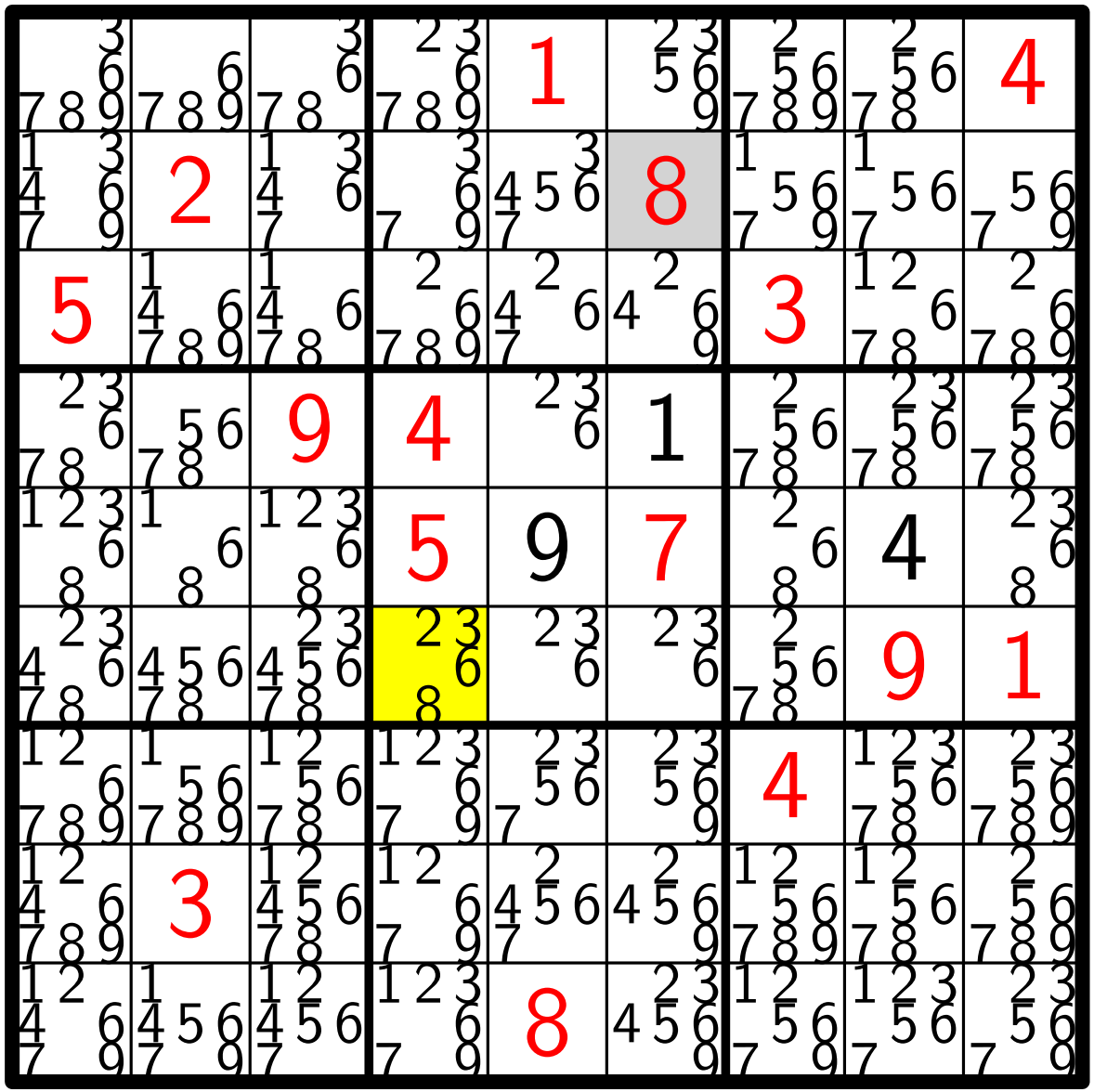
14

Sudoku 1



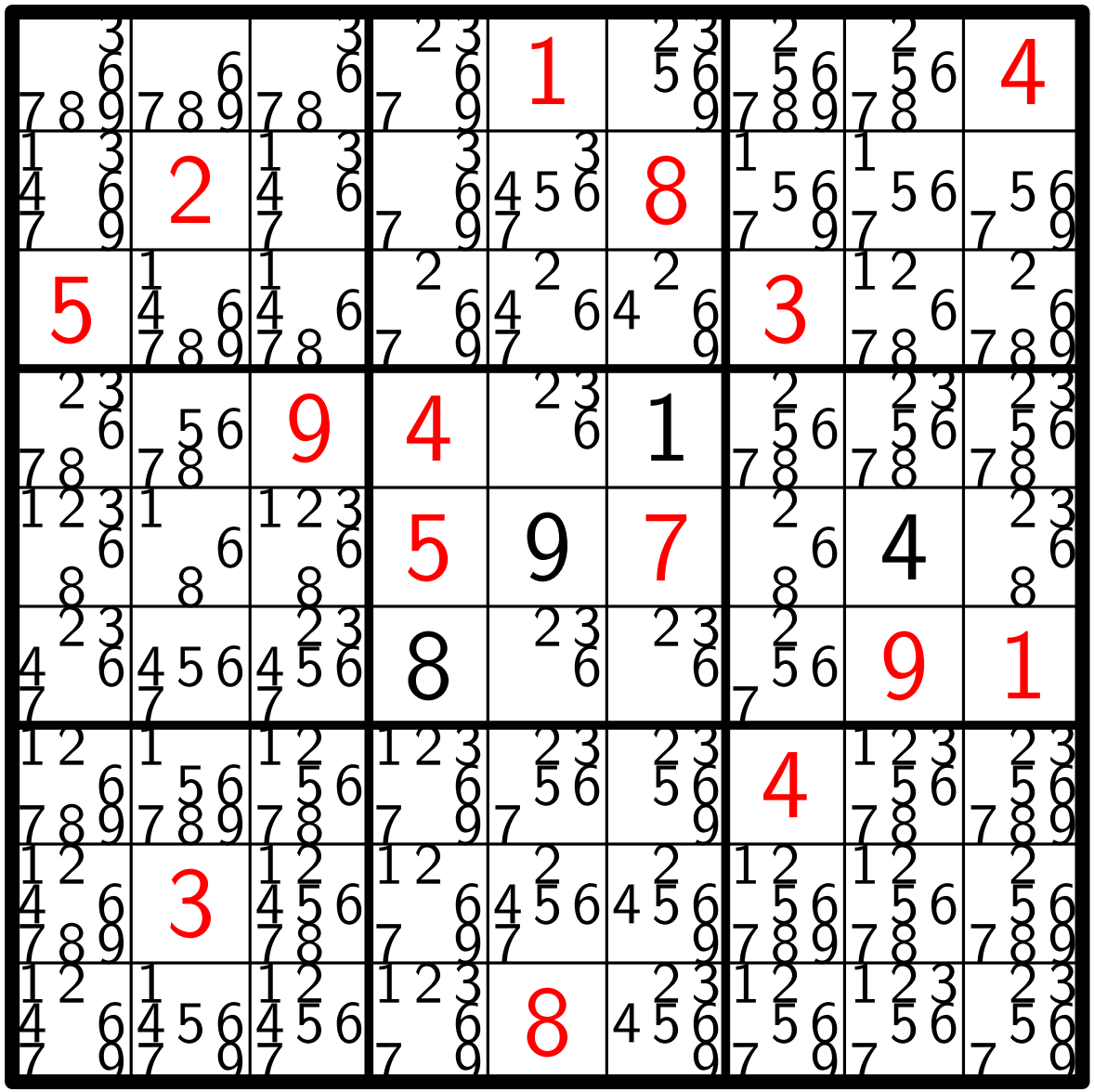
14

Sudoku 1



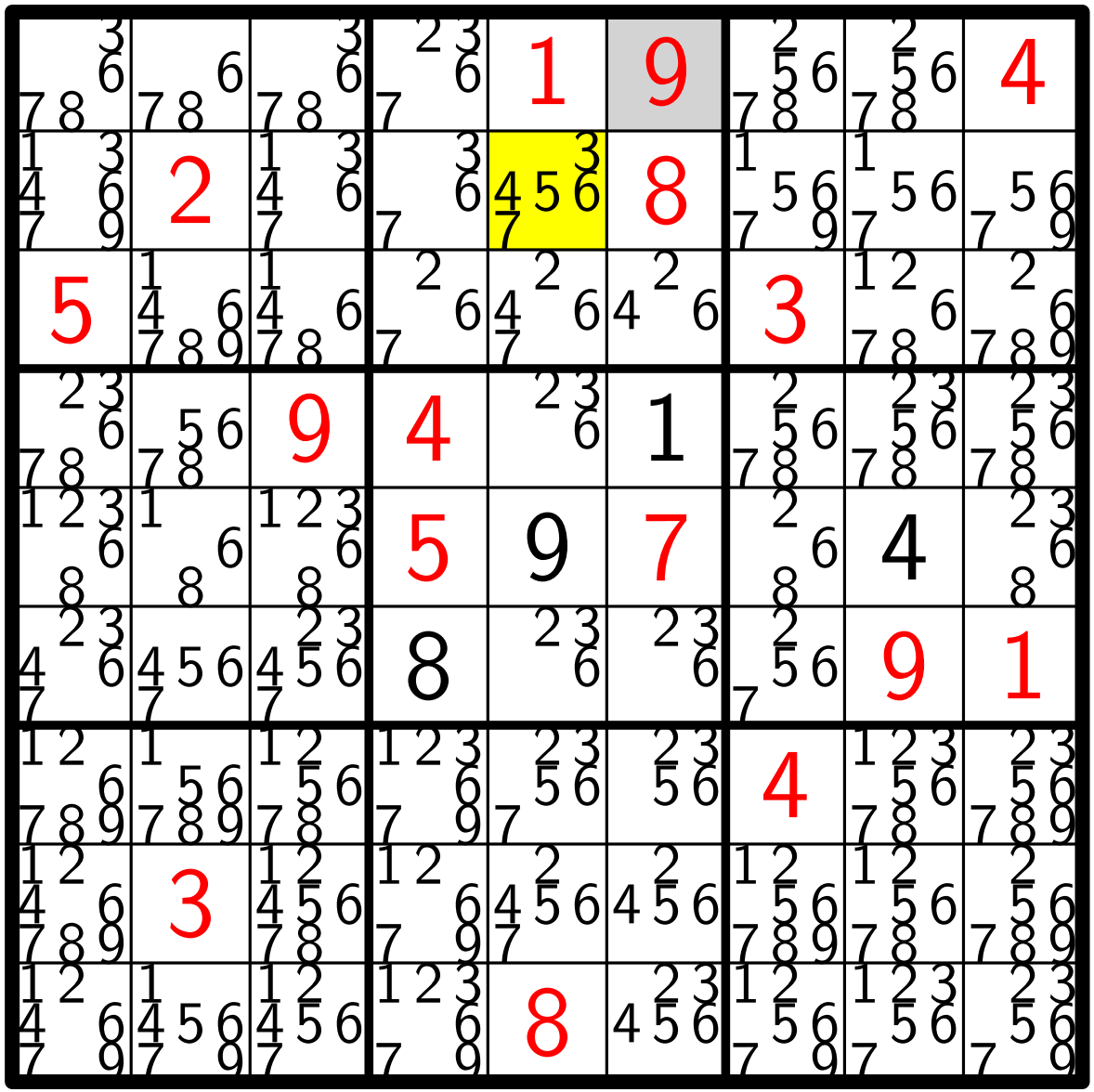
15

Sudoku 1



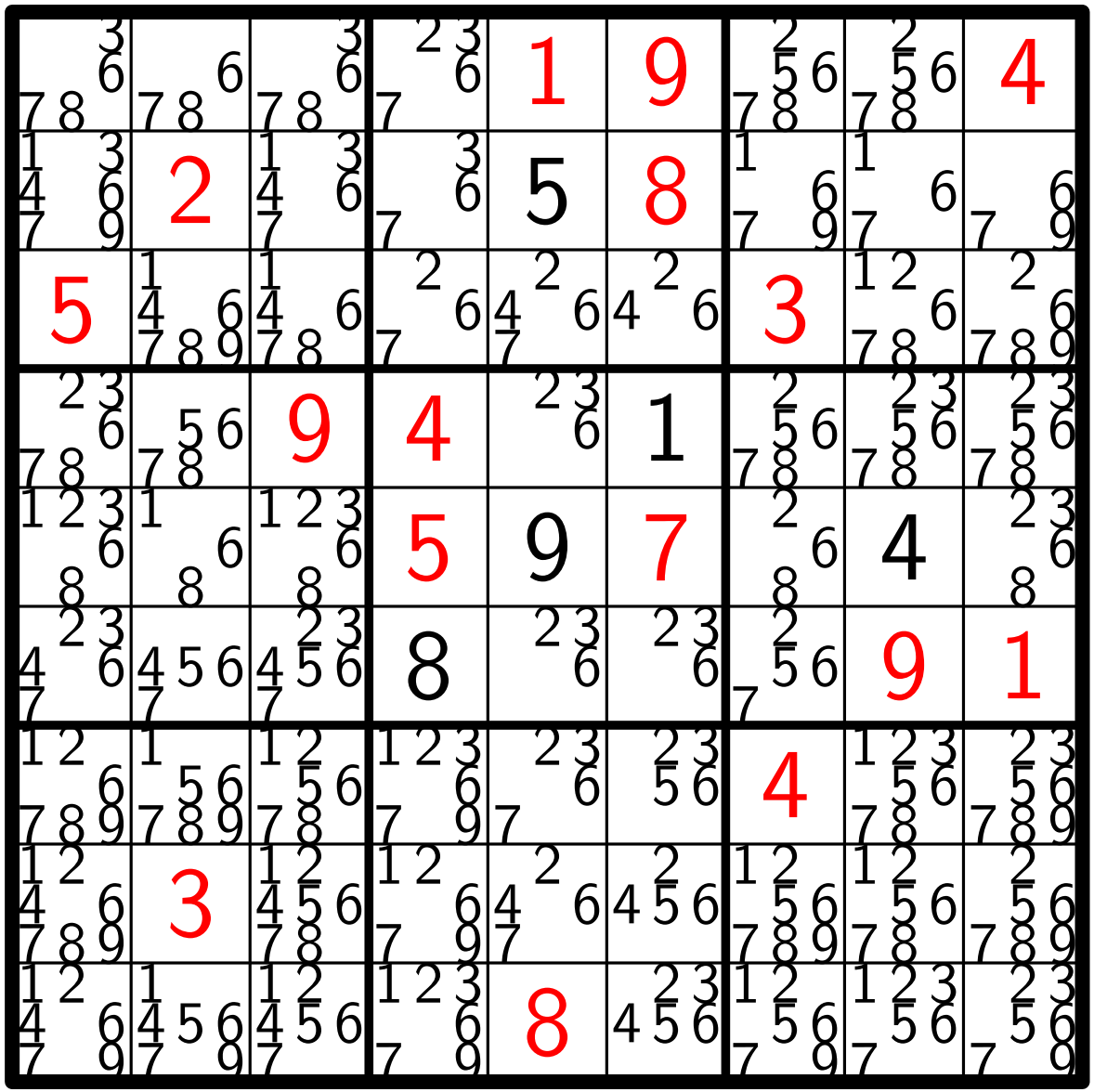
15

Sudoku 1



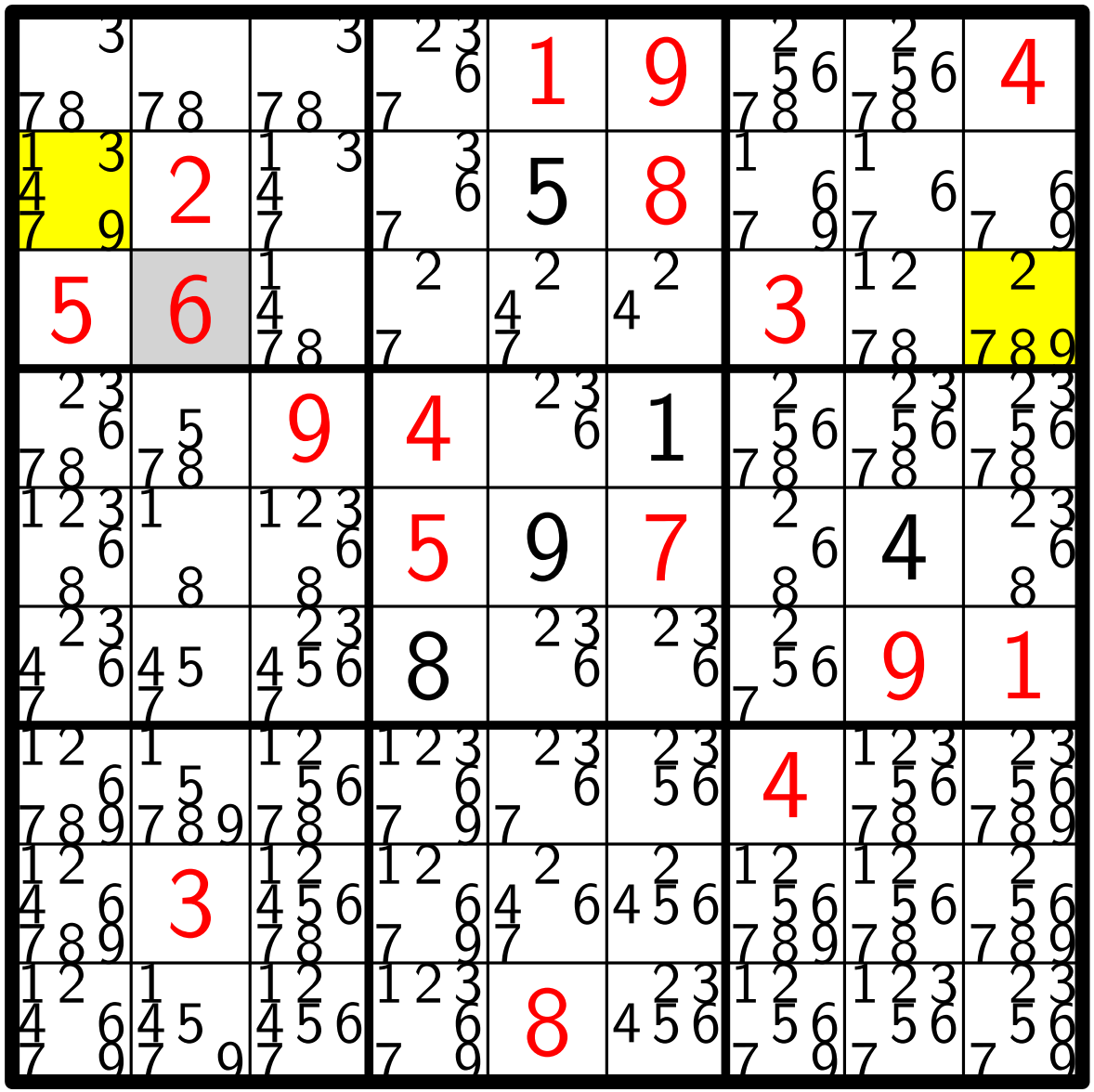
16

Sudoku 1



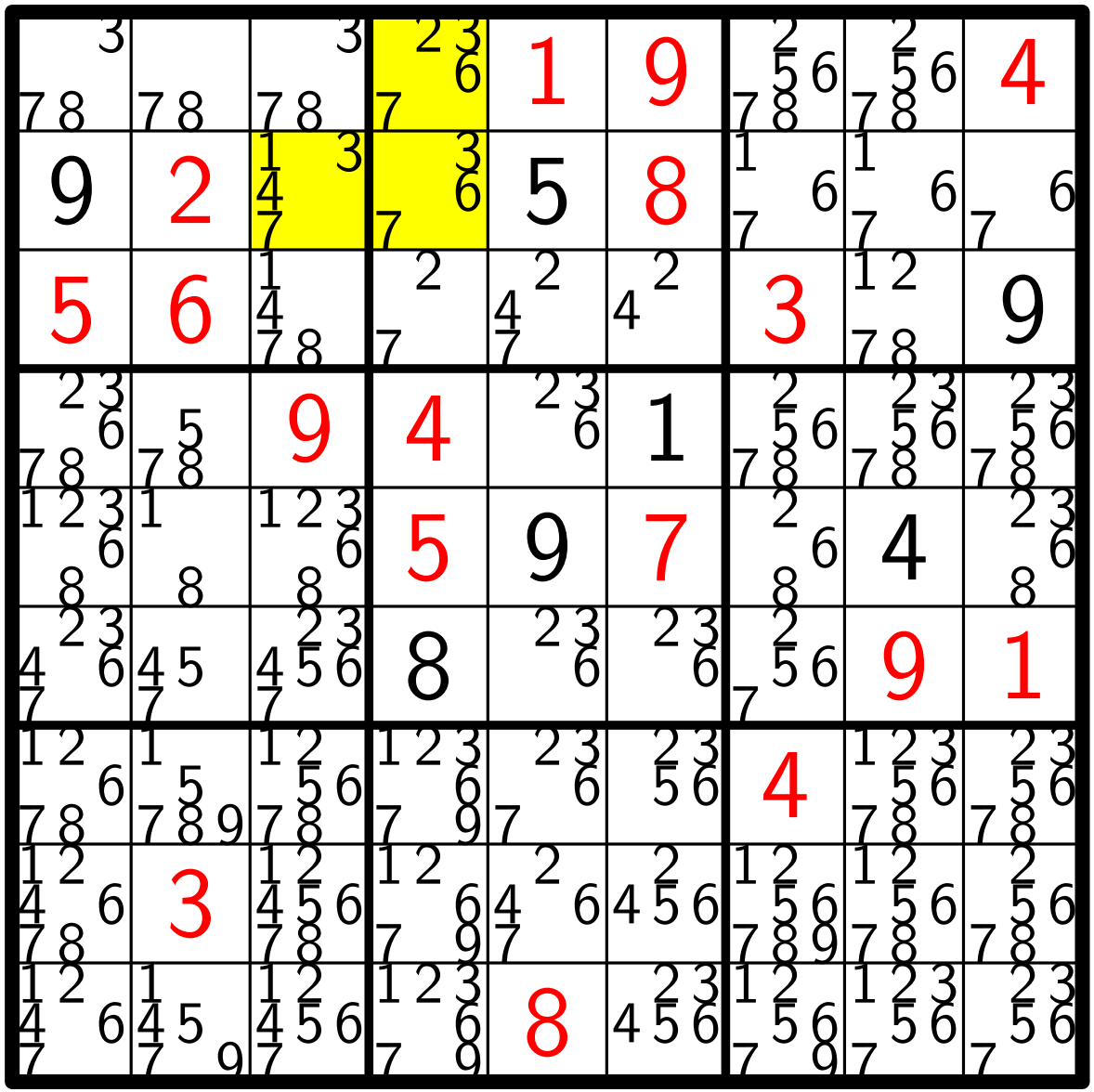
16

Sudoku 1



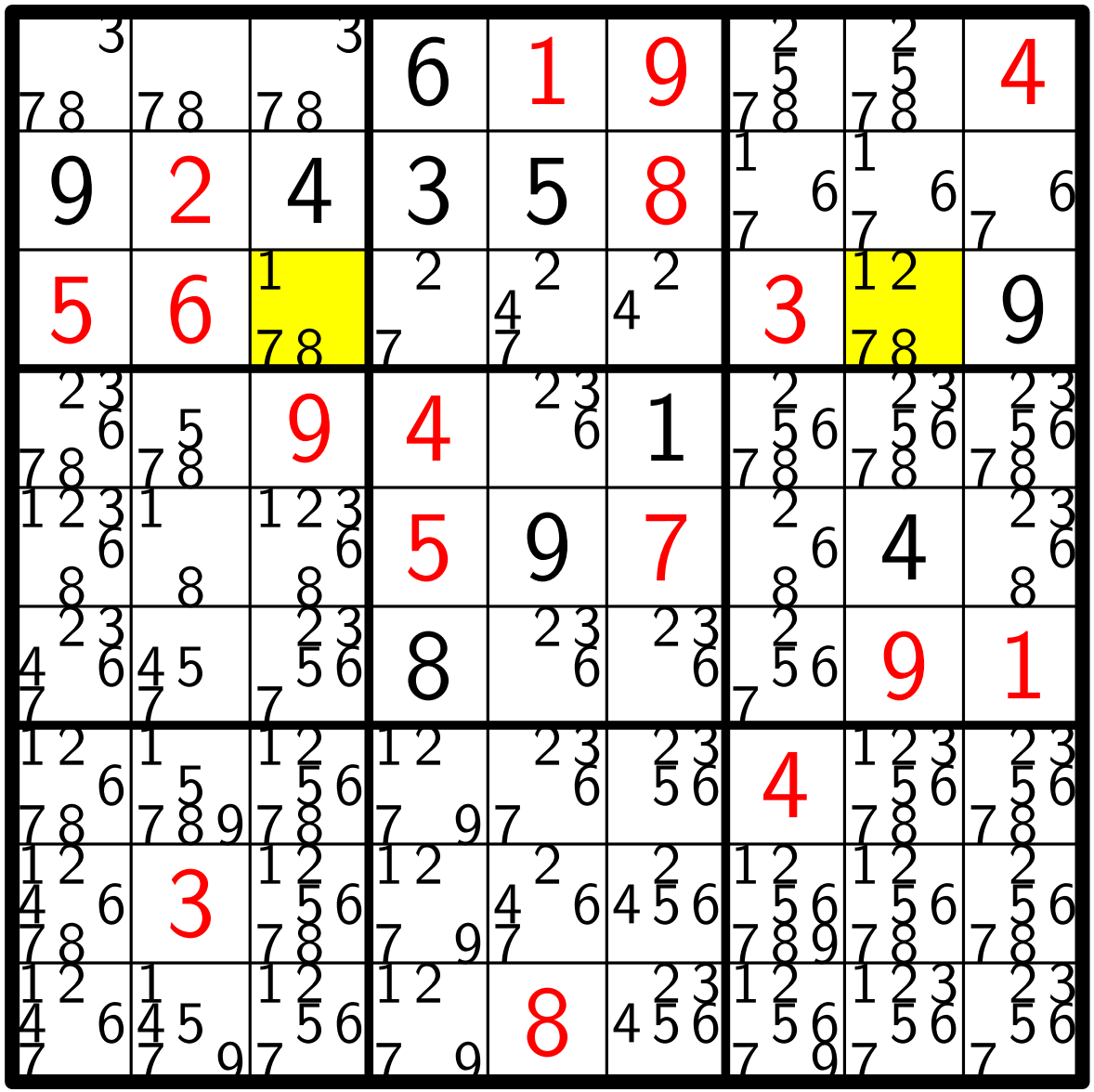
17

Sudoku 1



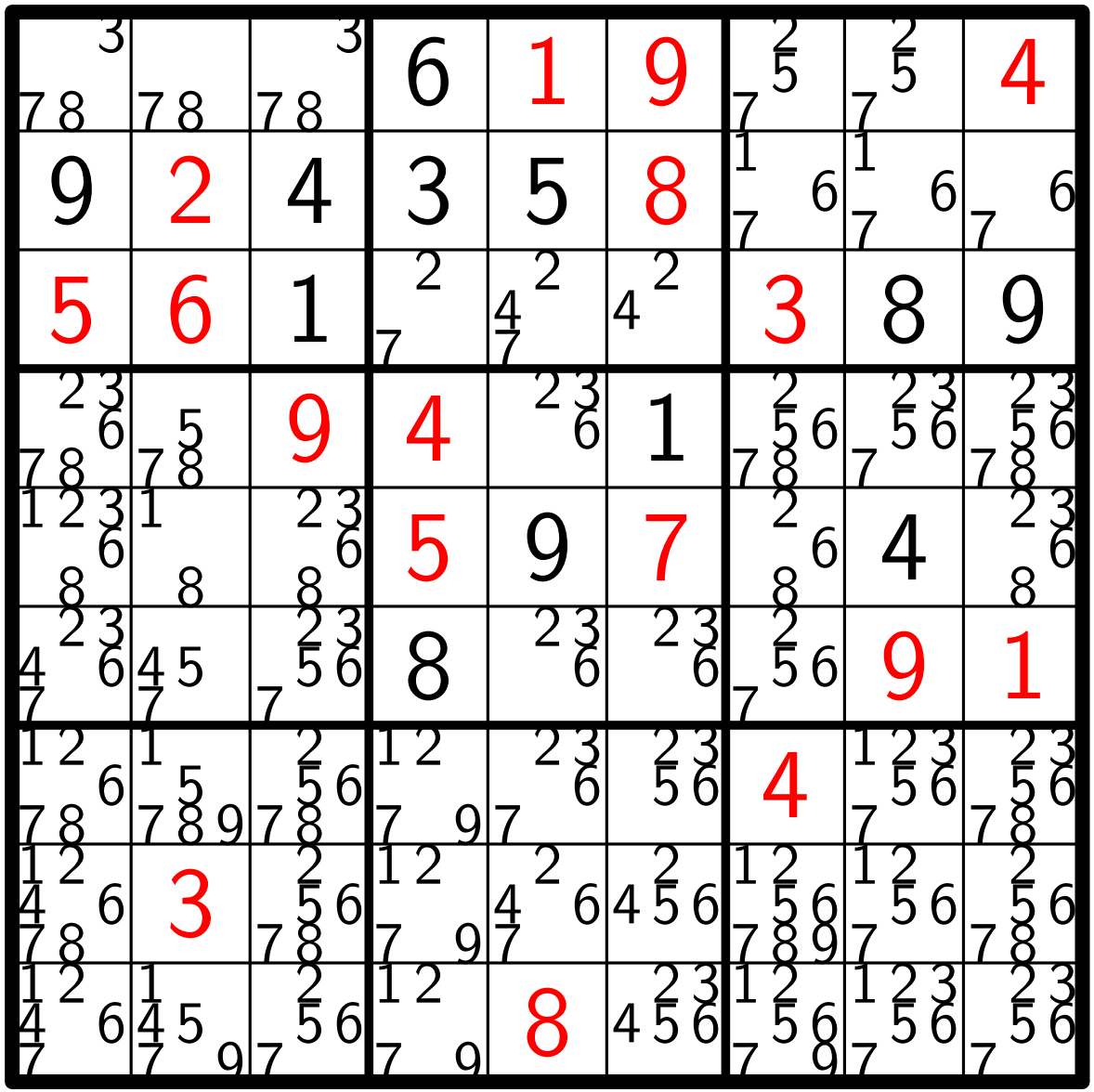
17

Sudoku 1

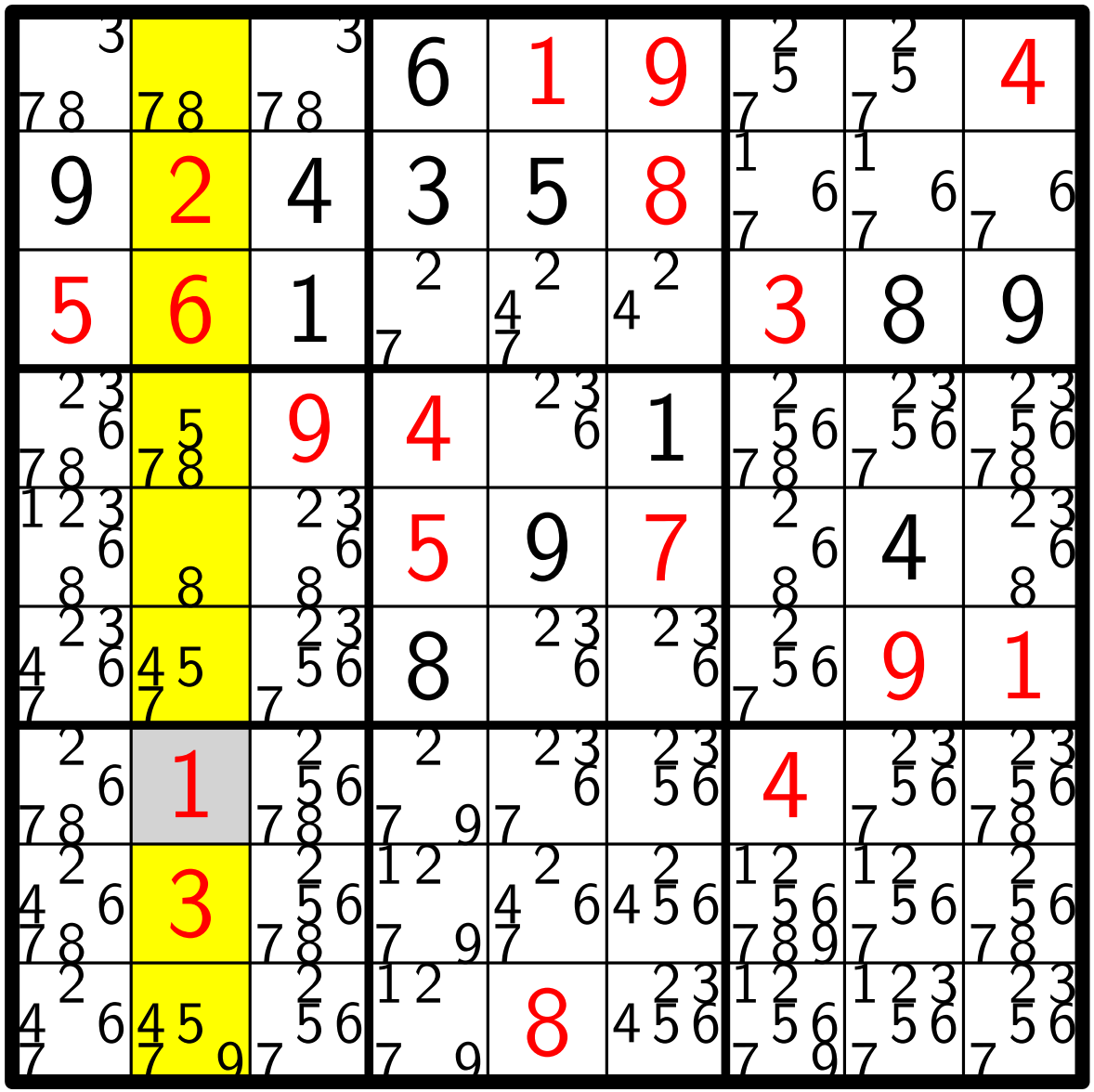


17

Sudoku 1

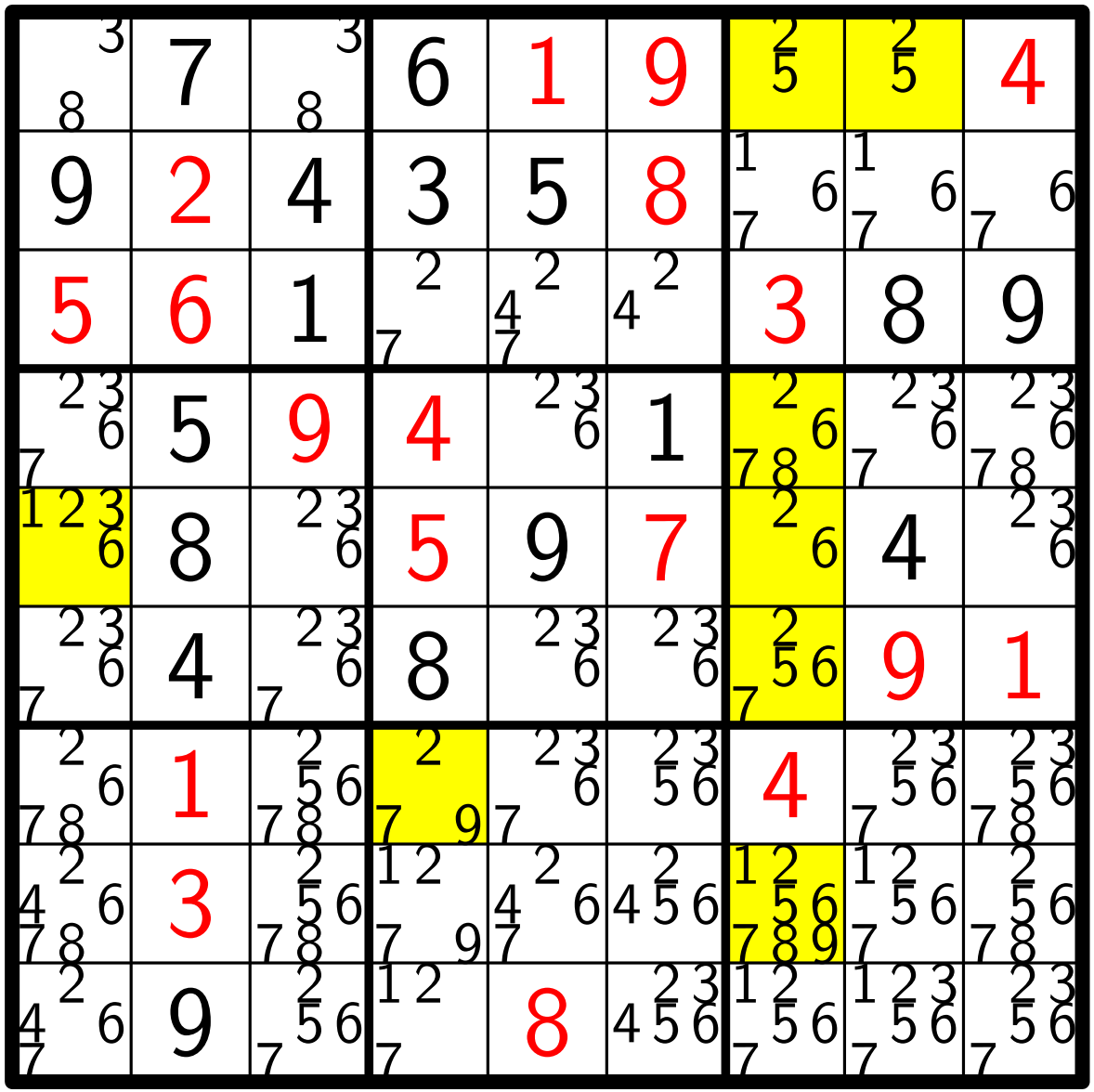


Sudoku 1



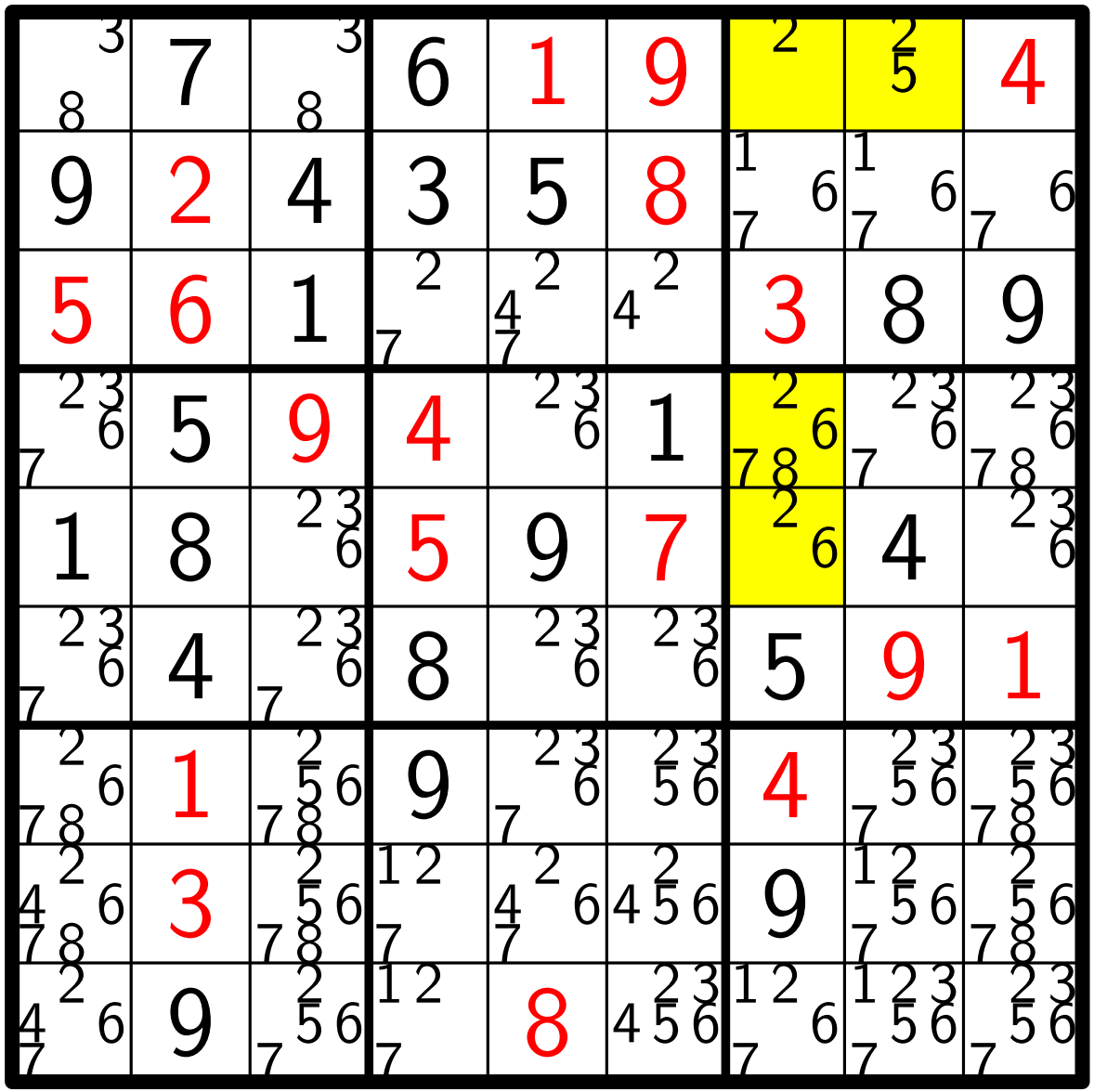
18

Sudoku 1



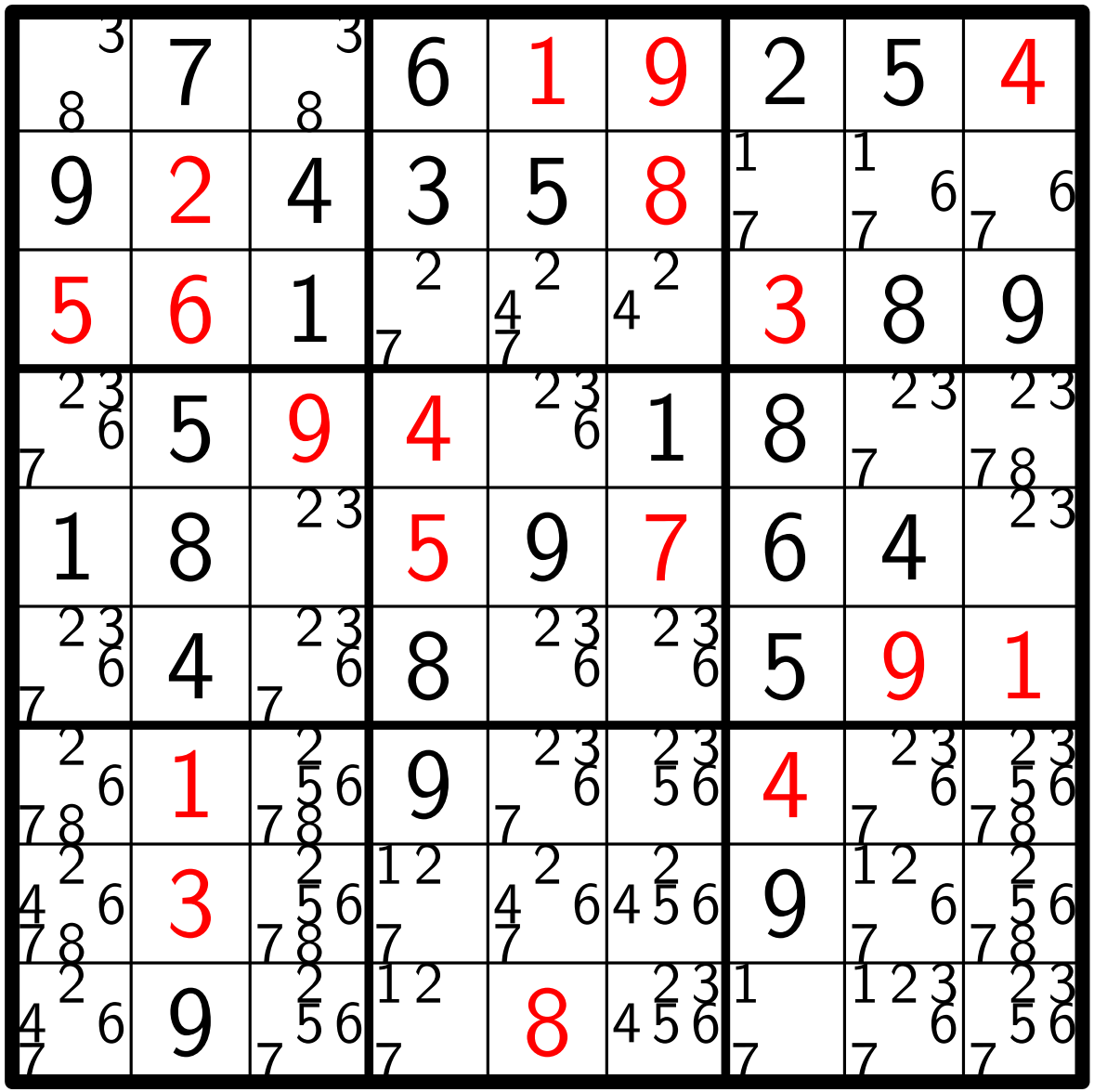
18

Sudoku 1


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

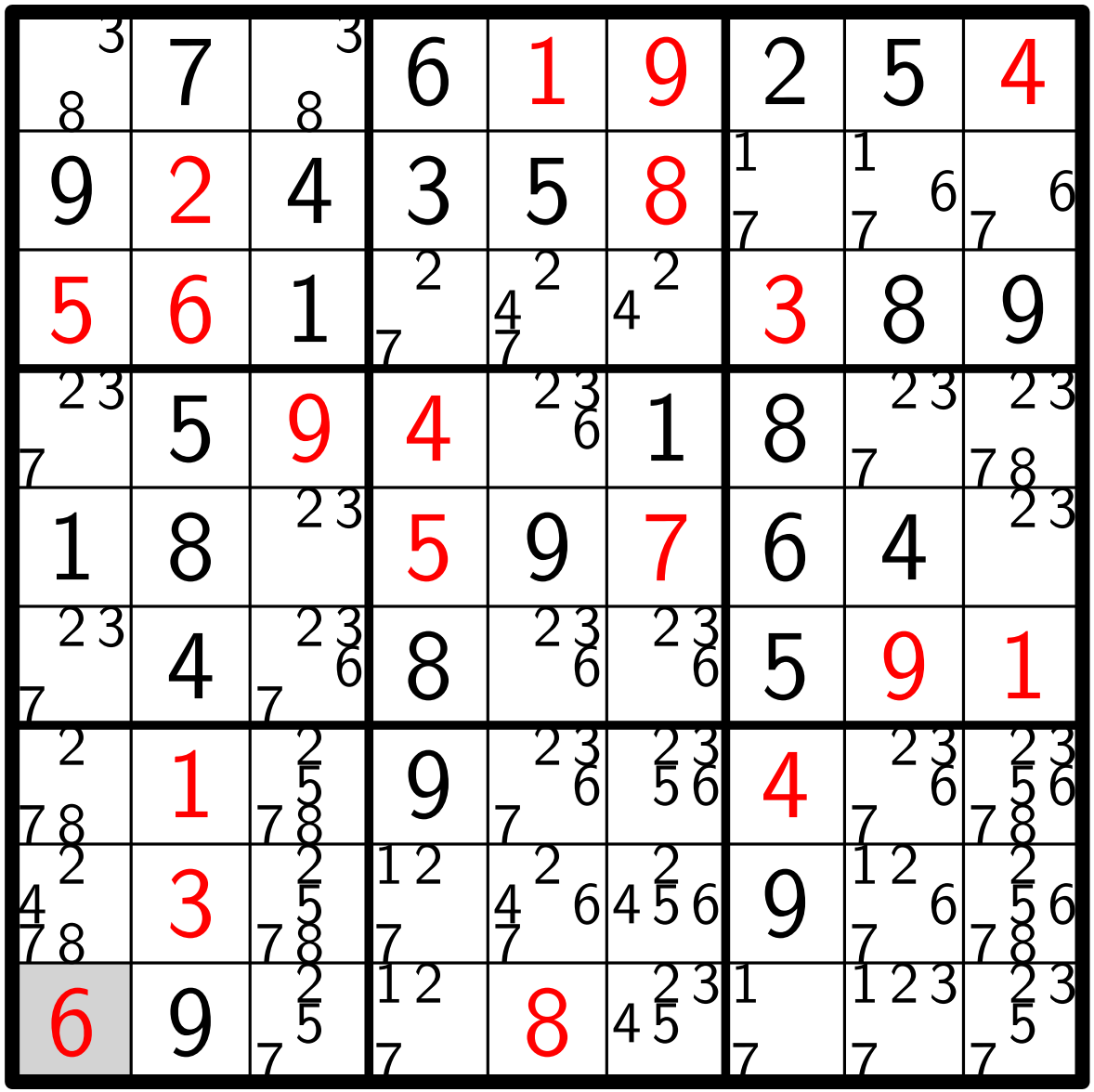
18

Sudoku 1


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

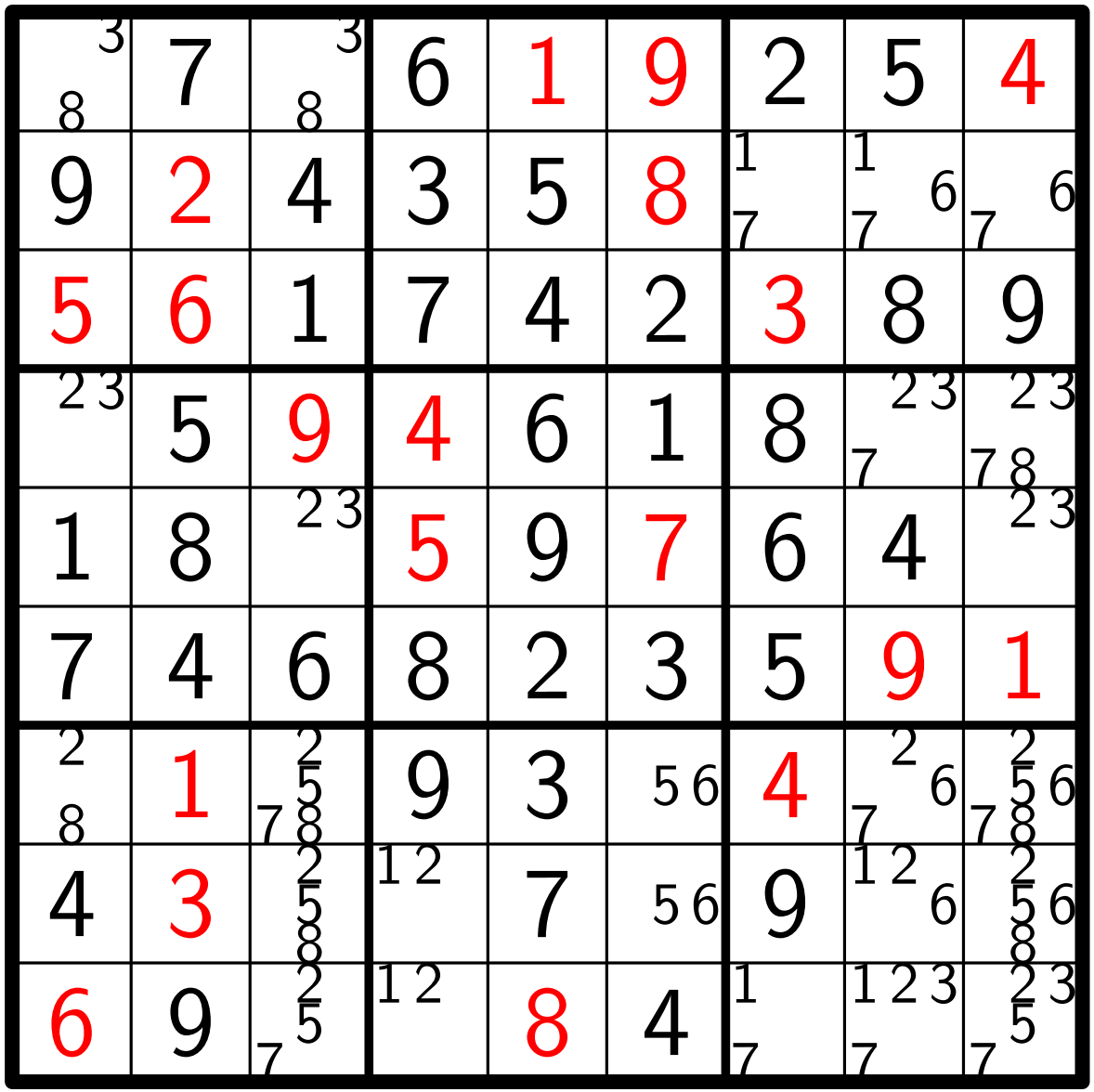
18

Sudoku 1


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

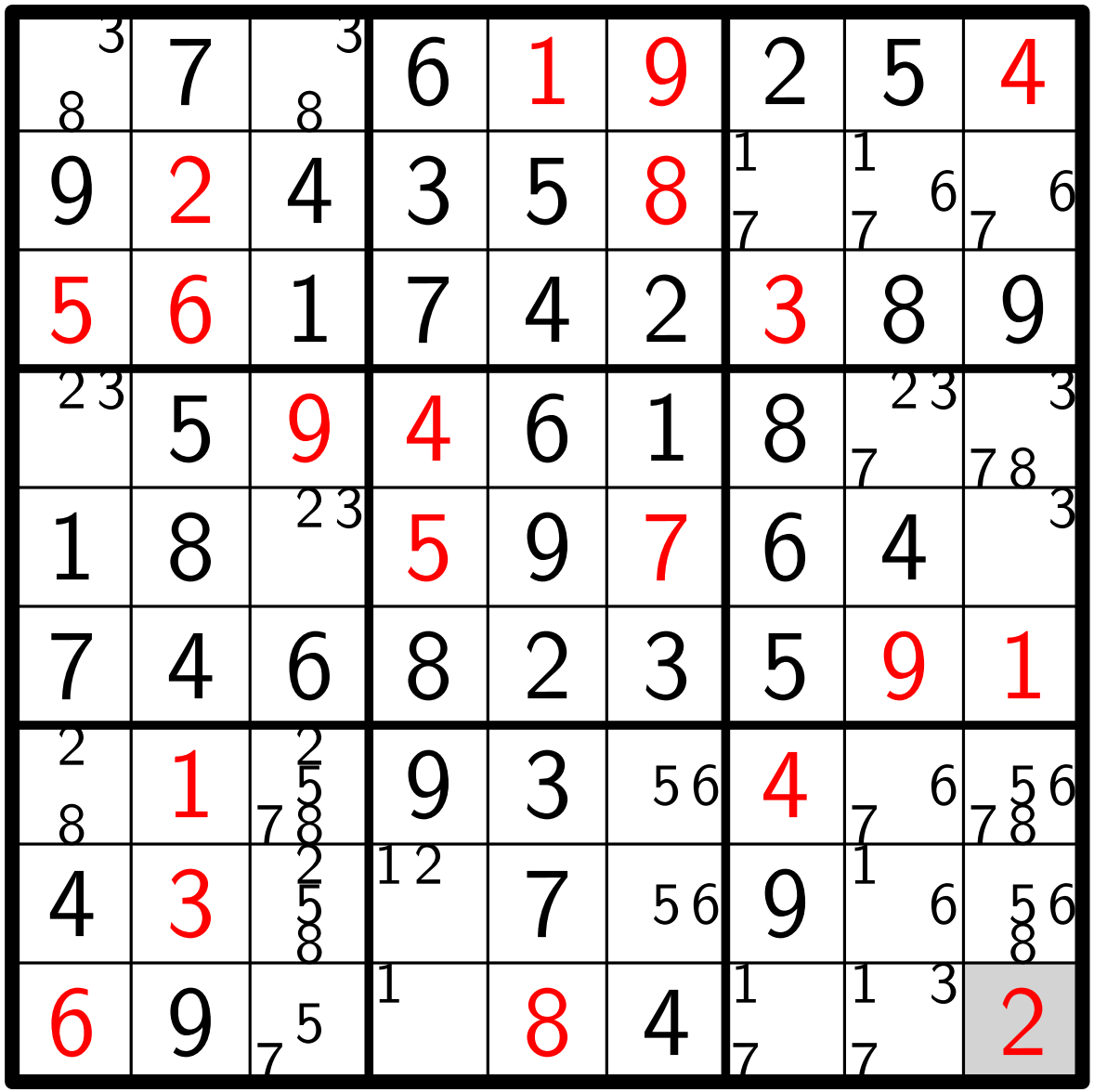
19

Sudoku 1


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

19

Sudoku 1


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

20

Sudoku 1

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

20

Sudoku Level 2

If FIX-SQUARE() does not solve the sudoku, we have to use a version of backtracking

Sudoku Level 2

If FIX-SQUARE() does not solve the sudoku, we have to use a version of backtracking

- Choose a square sq with a small number of still valid numbers.
- For all valid numbers nr, fix the square sq with this number by calling FIX-SQUARE(sq, nr)
- If the answer is "No valid solution" choose next valid number
- Otherwise we have either a solution, or we need to backtrack recursively for another square

Sudoku Level 2

If FIX-SQUARE() does not solve the sudoku, we have to use a version of backtracking

- Choose a square sq with a small number of still valid numbers.
- For all valid numbers nr, fix the square sq with this number by calling FIX-SQUARE(sq, nr)
- If the answer is "No valid solution" choose next valid number
- Otherwise we have either a solution, or we need to backtrack recursively for another square

The number of recursively nested backtracking steps determines the level of a Sudoku. Most of the difficult Sudokus have just 1 or 2 such steps (Level 2 or Level 3).

Example Level 2 Sudoku

24 givens:

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

Example Level 2 Sudoku

24 givens:

Inserting all givens with FIX-SQUARE() does not solve the sudoku

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

Example Level 2 Sudoku

24 givens:

Inserting all givens with FIX-SQUARE() does not solve the sudoku

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

Backtracking on a fixed square with small number of possible entries

Example Level 2 Sudoku

24 givens:

Inserting all givens with FIX-SQUARE() does not solve the sudoku

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

Backtracking on a fixed square with small number of possible entries

6

Solves the sudoku

Example Level 2 Sudoku

24 givens:

Inserting all givens with FIX-SQUARE() does not solve the sudoku

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

Backtracking on a fixed square with small number of possible entries

6

Solves the sudoku

Example Level 2 Sudoku

24 givens:

Inserting all givens with FIX-SQUARE() does not solve the sudoku

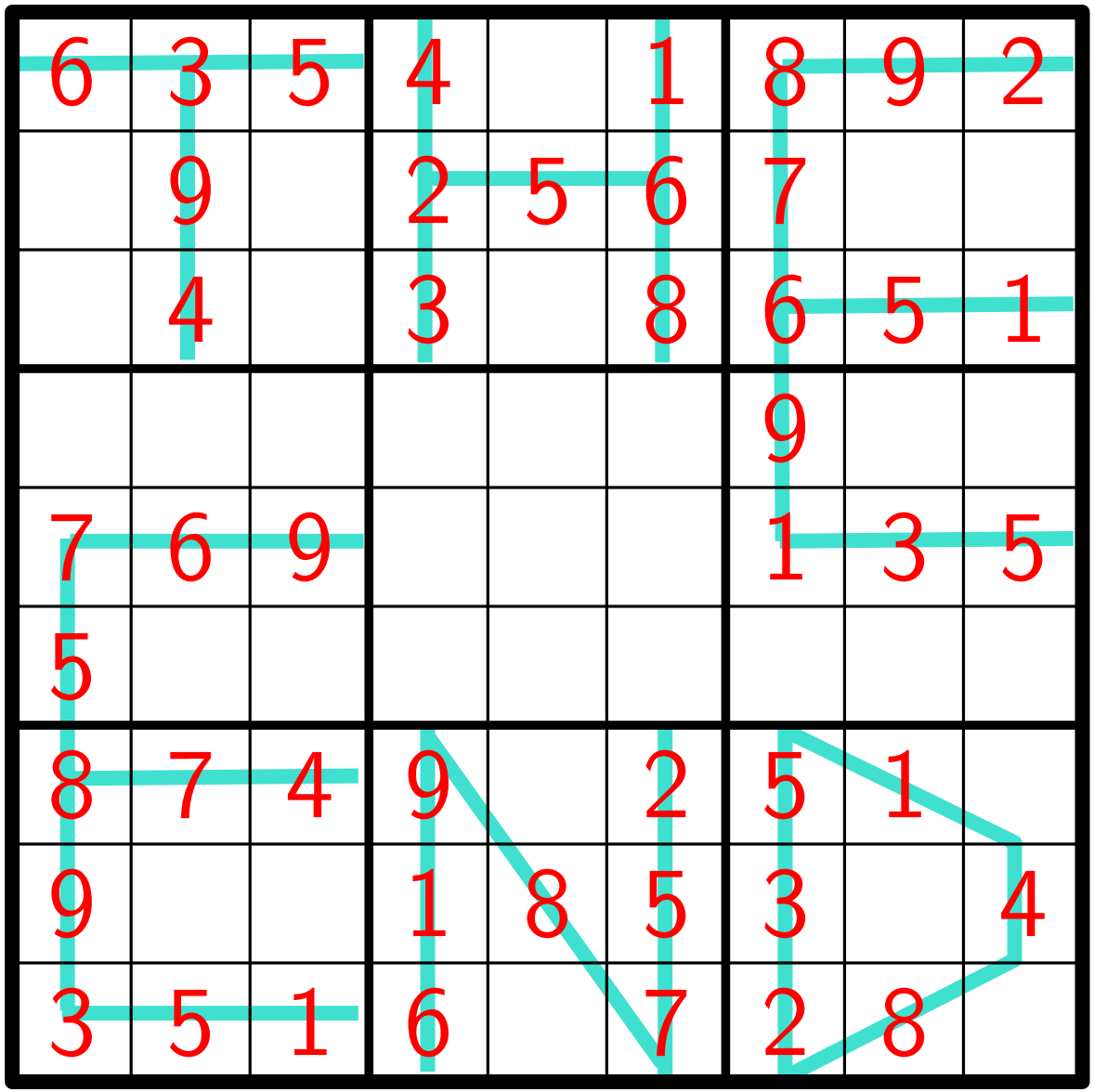
8	3	5	4		1		9	2
	9			5				3
	4	7	3	9			5	1
			5		3	9	6	
7		9				1	3	5
5		3			9			
	7	8		3		5	1	6
	2			8	5	3	7	4
3	5		6		7			9

Backtracking on a fixed square with small number of possible entries

6 Solves the sudoku

8 leads to a contra-diction

The End



The End

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