Sudoku Solver

11.06.2018

Magson Gao, Shreeyam Kacker

Sudoku Primer

Project Description

What is Sudoku?

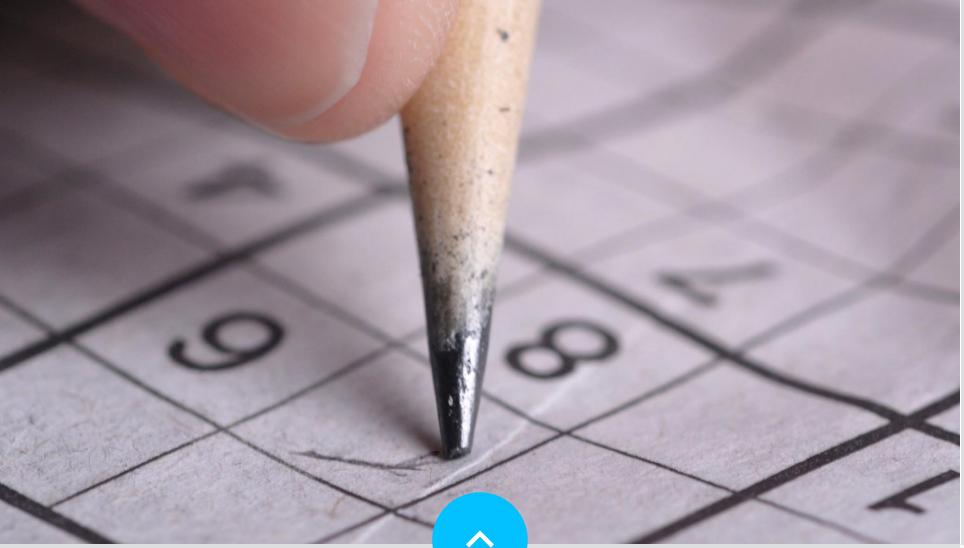
Fill 9x9 Grid with 81 × 1-9

Why FPGA?

Parallelism

Techniques?

Candidate Line, Naked and Hidden Groups

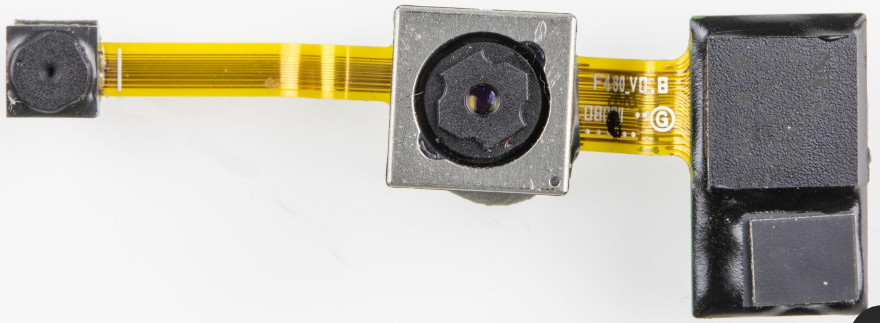


Human Techniques

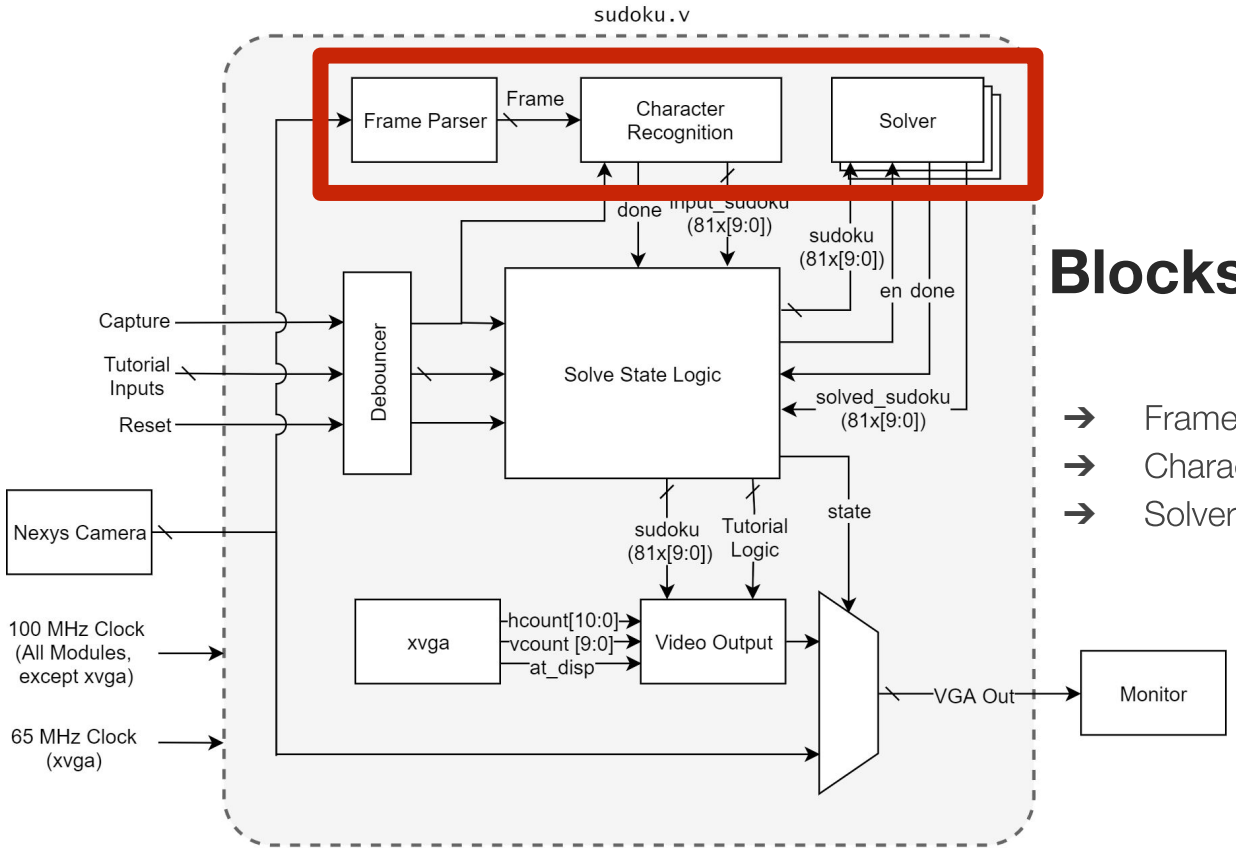
- ➔ Single position and single candidate
- ➔ Candidate line
- ➔ Hidden group
- ➔ Many more… (but only we have 8 minutes)

Goals

- ➔ Solve Sudoku using human techniques
- ➔ Read Sudoku using a camera
- ➔ Stretch Goal: include a tutorial mode which displays the steps for any solve



Block Diagram



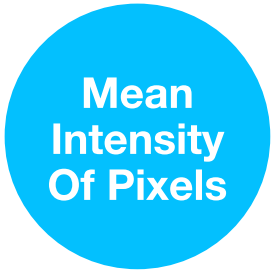
Blocks of Interest

- ➔ Frame Parser
- ➔ Character Recognition
- ➔ Solver

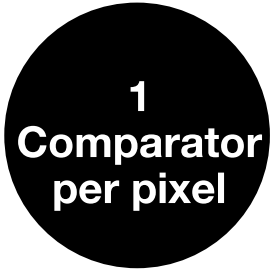
Frame Parsing

Reducing the number of gates for Character Recognition

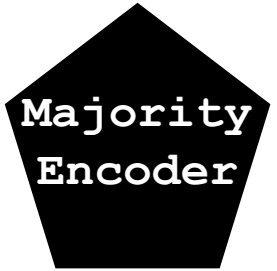
Calculate Threshold



Convert to Black and White



Resize



Character Recognition

Template Matching

Store flattened B/W values of #s

ROM size: 9× pixels/char (bits)

Compare with processed frames

9 XOR operations/cycle
9 result registers

Select

Comparison operation using LUTs

Solving a Sudoku

Keywords

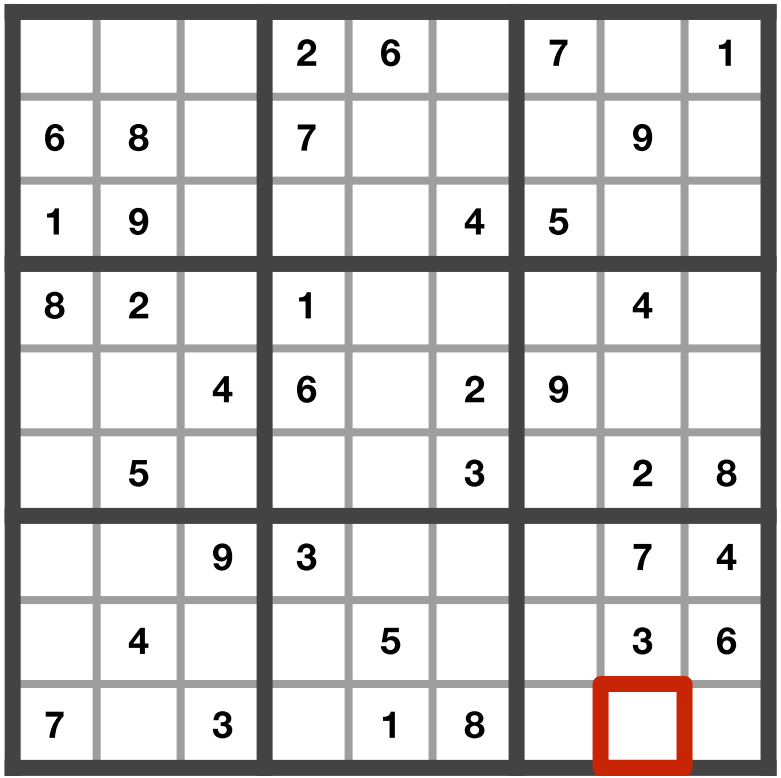
			2	6		7		1
6	8		7				9	
1	9				4	5		
8	2		1				4	
		4	6		2	9		
	5				3		2	8
		9	3				7	4
	4			5			3	6
7		3		1	8			

Squares

Rows

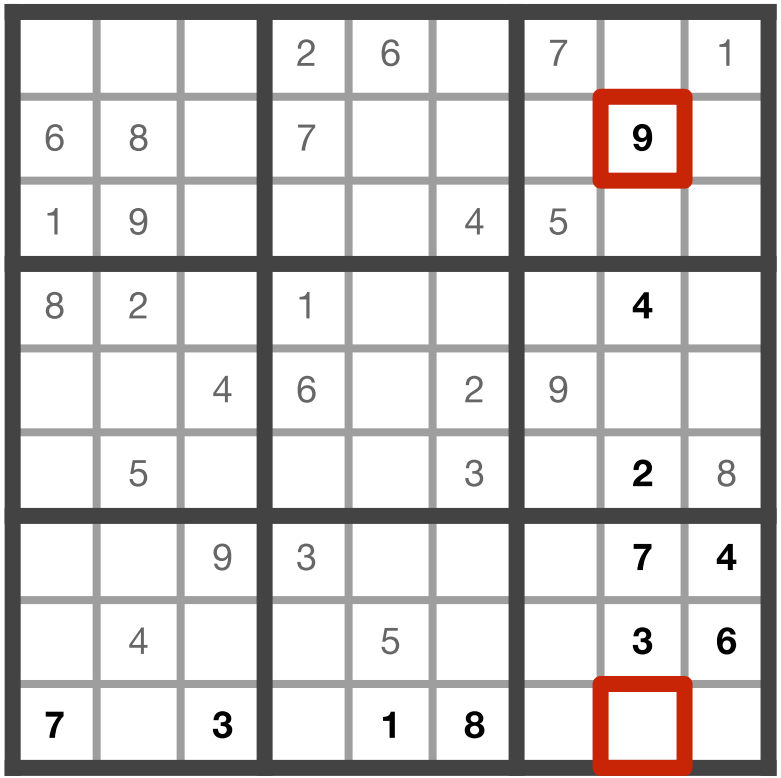
Columns

Single Candidate/Position



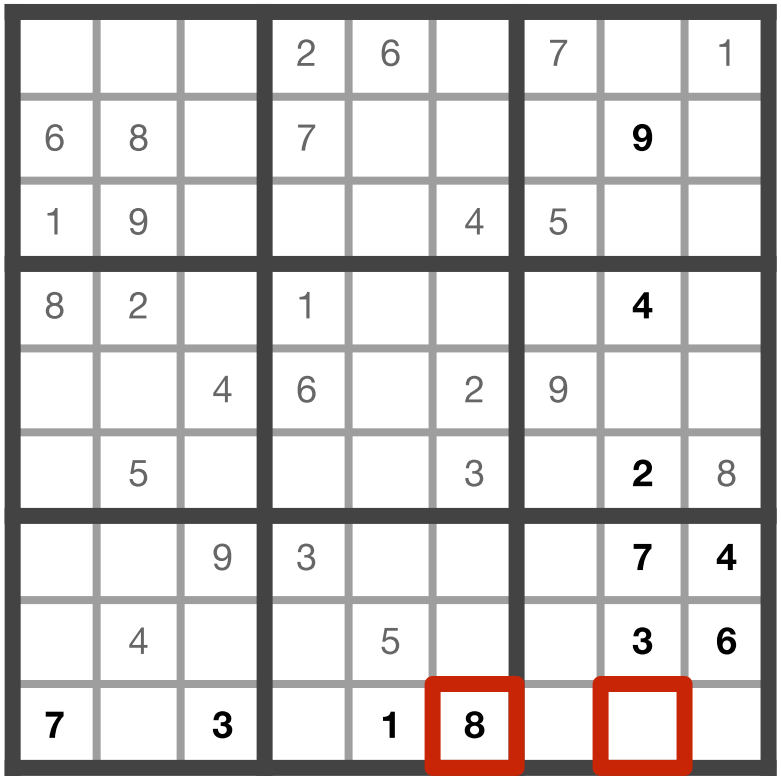
What can go here?

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



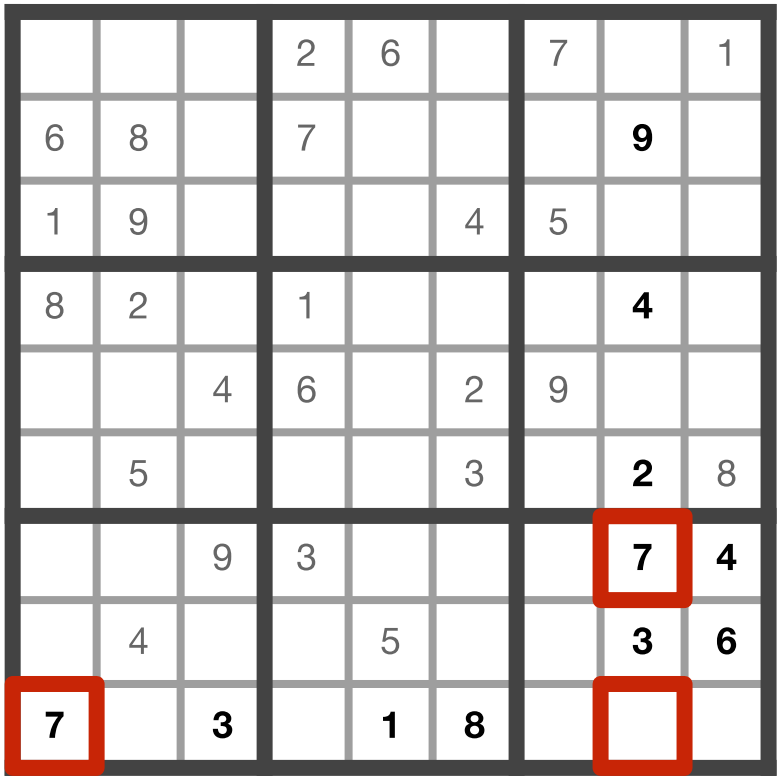
What can go here?

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



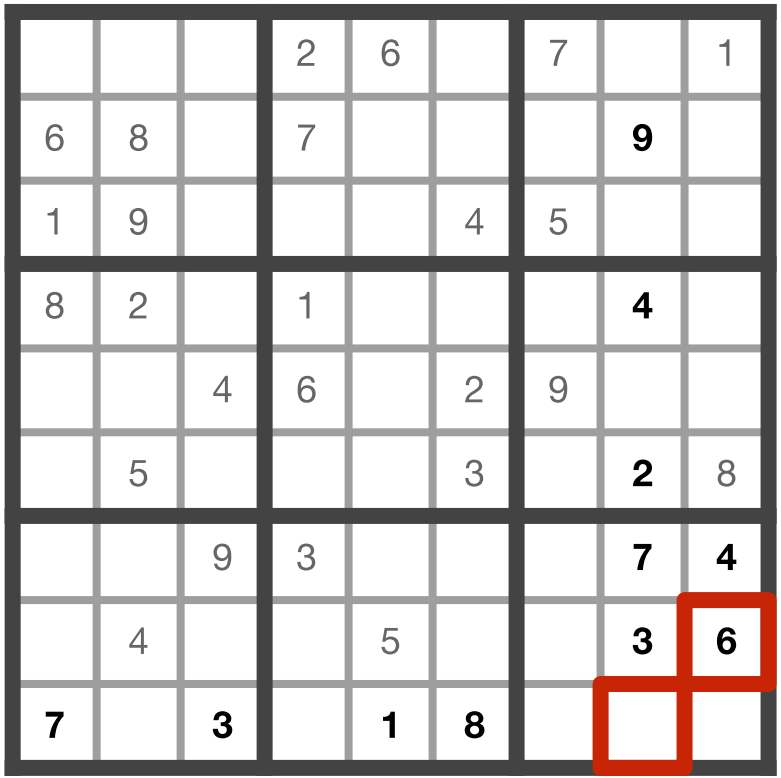
What can go here?

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



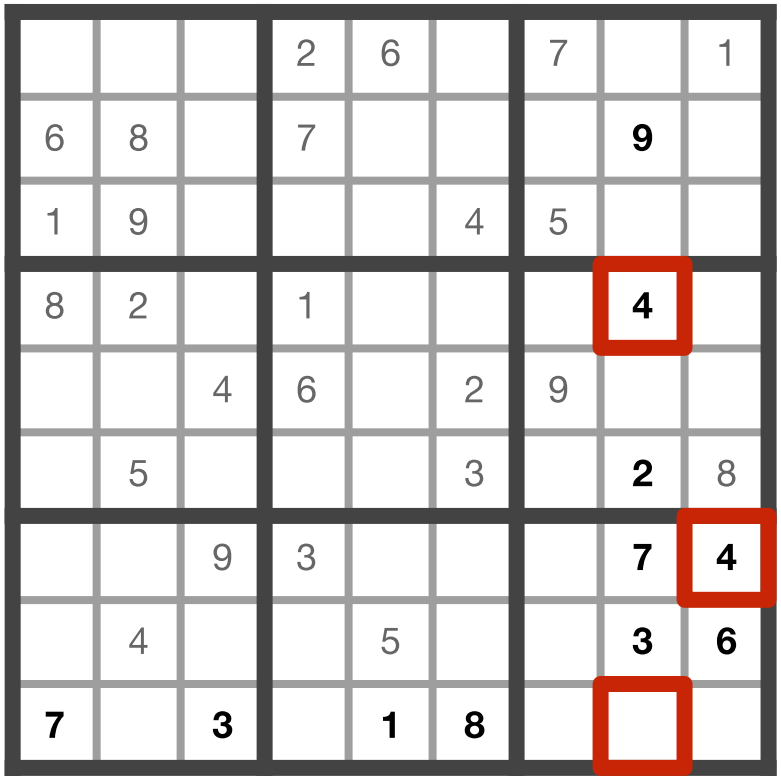
What can go here?

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



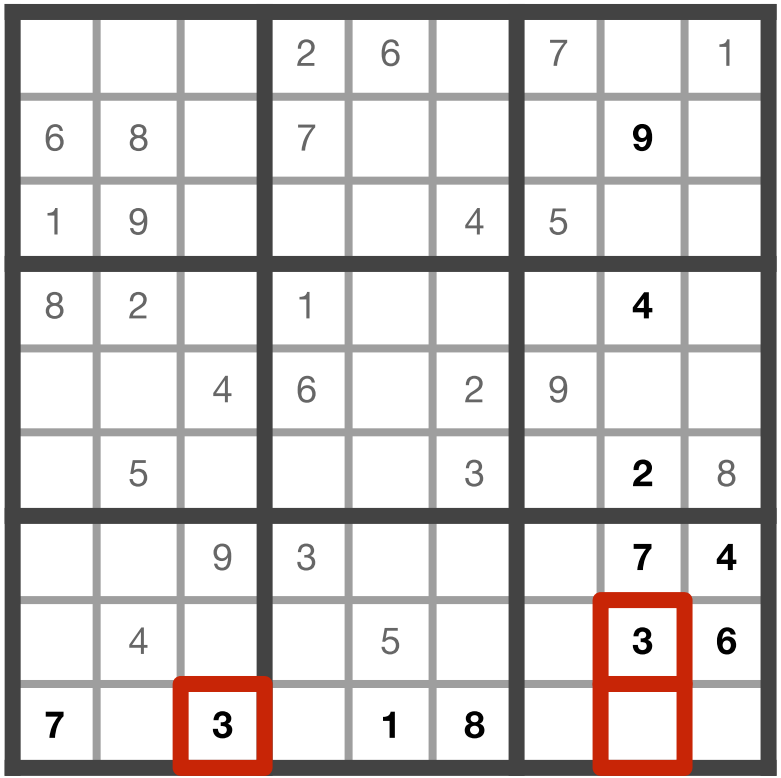
What can go here?

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



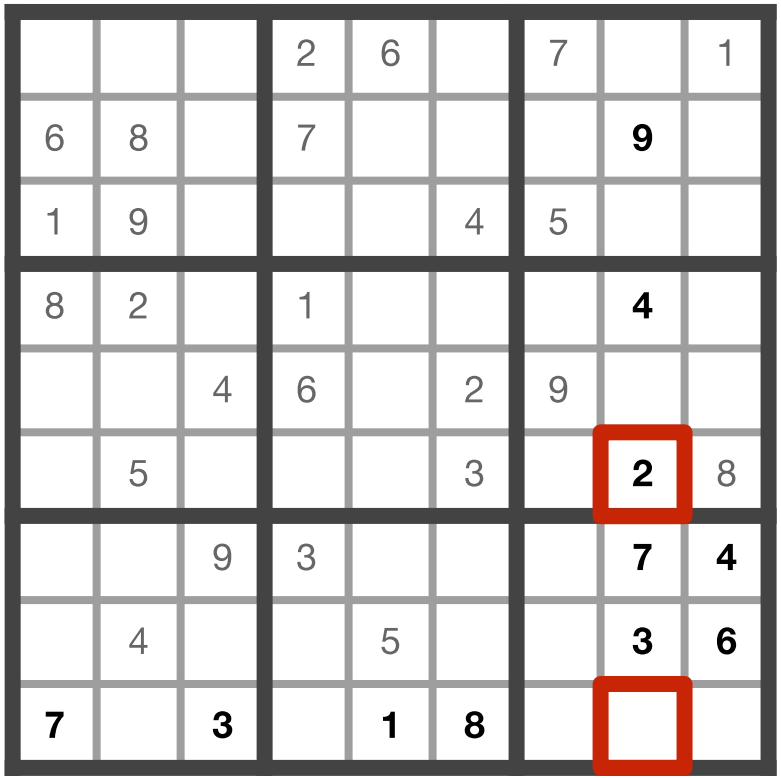
What can go here?

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



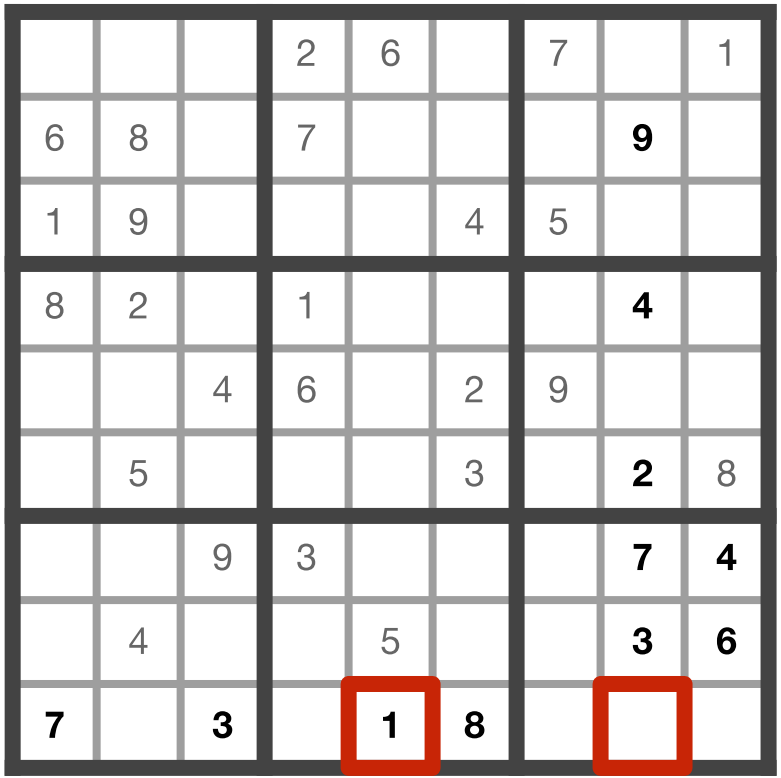
What can go here?

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



What can go here?

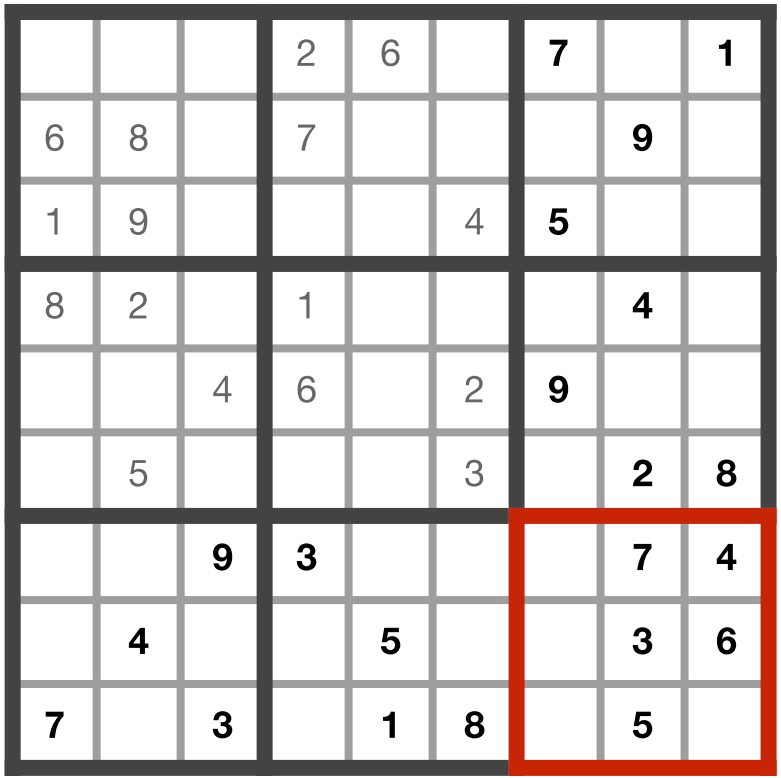
1,2,3,4,5,6,7,8,9



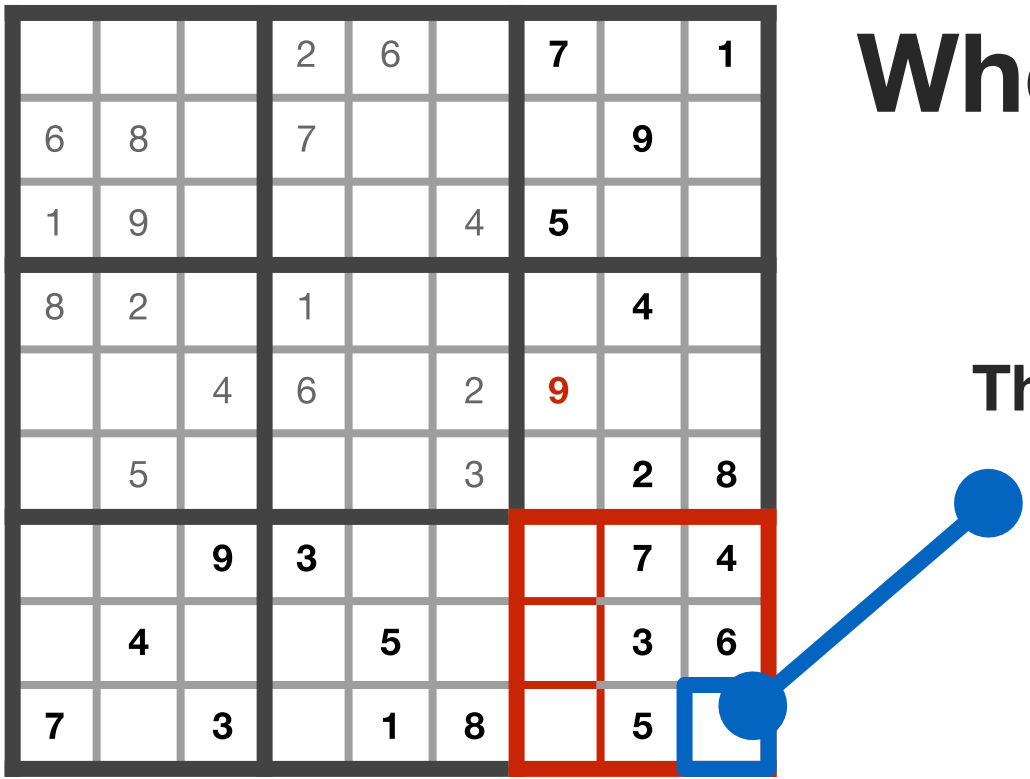
What can go here?

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

5 is the Single Candidate



Where can 9 go?



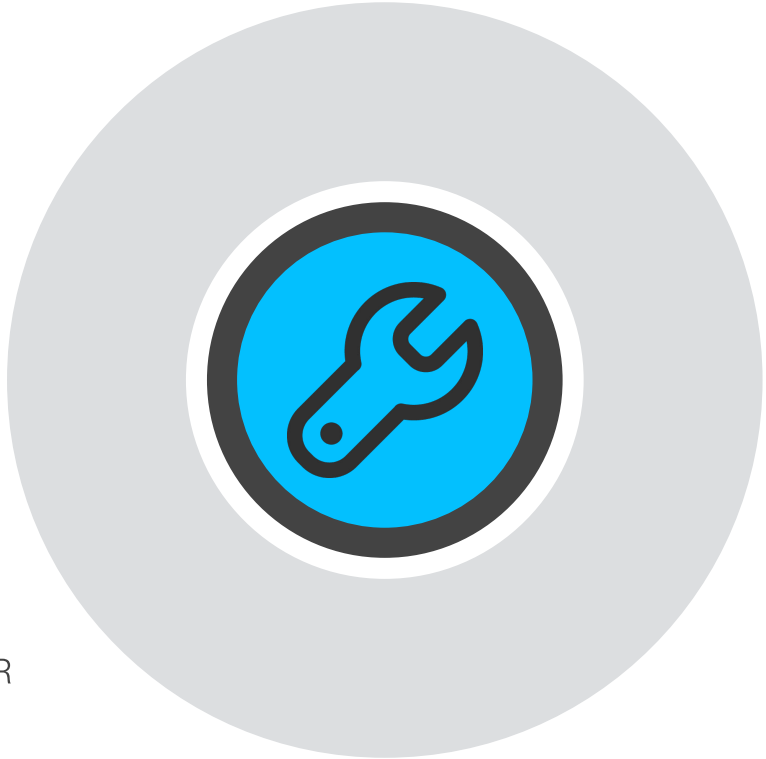
Where can 9 go?

This is the Single Position

How can a bunch of gates do Sudoku?

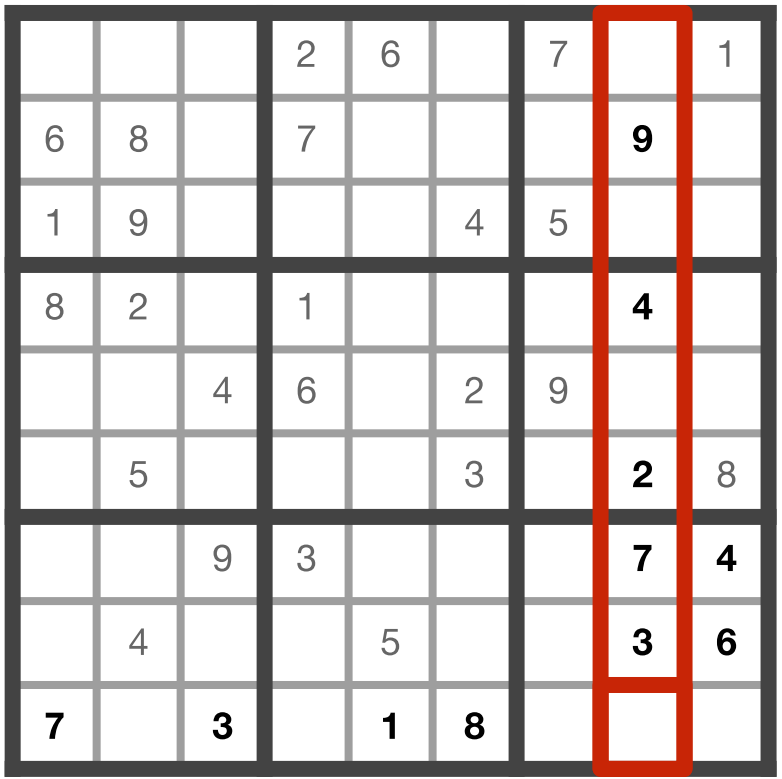
One-hot Representation

- ➔ Simplifies logic significantly
- ➔ Empty Cells = 0
- ➔ Determining single candidates is a simple AND operation with masks of possible values
- ➔ Determining single positions is a NOR and AND operation between masks



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

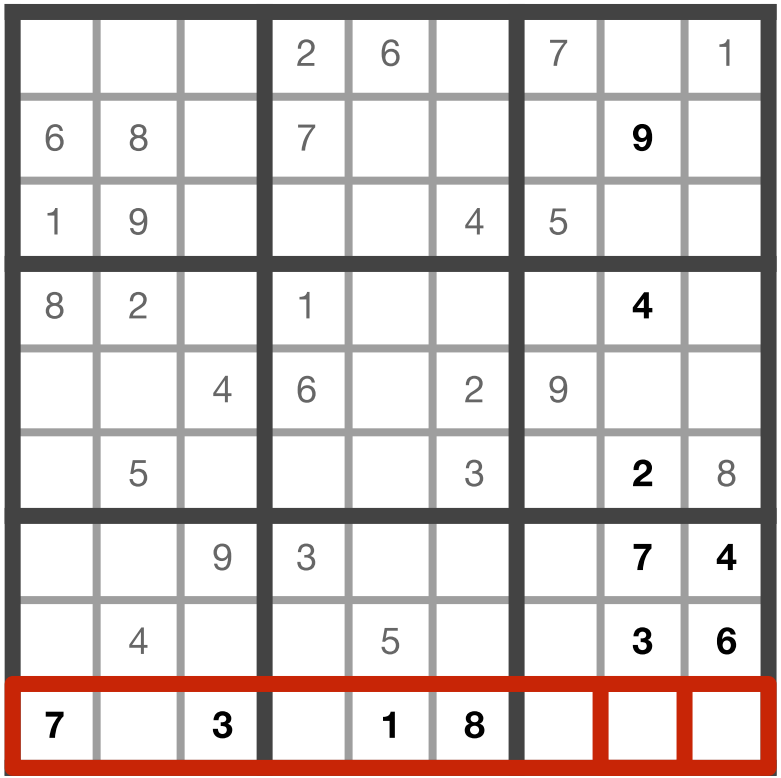
What can go here?



What can go here?

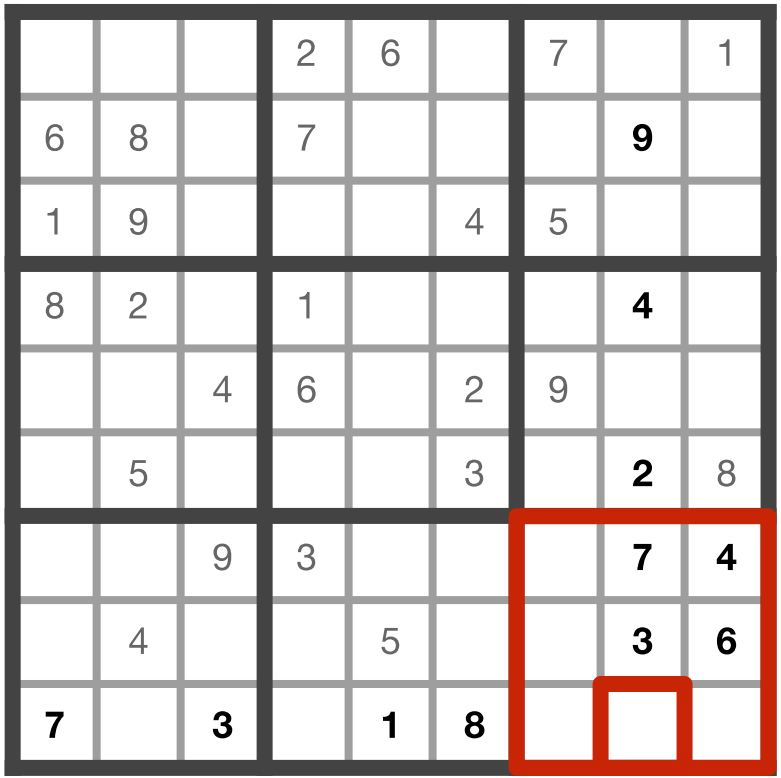
```
    987654321
9= 100000000
4= 000001000
2= 000000010
7= 001000000
3= 000000100
|= 101001110
~= 010110001
```

8,6,5 or 1



What can go here?

```
    987654321
7=  001000000
3=  000000100
1=  000000001
8=  010000000
|=  011000101
~=  100111010   not 1 and not 8
 &  010110001
=   000110000
```



What can go here?

```
    987654321
7=  001000000
4=  000001000
3=  000000100
6=  000100000
|=  001101100
~=  110010011
 &  000110000
```

Must be a 5

Hidden Group Setup

- ➔ Suppose we have **N** numbers which can **only** be placed in **N** positions
- ➔ Then **all N positions must contain one of the N numbers**.

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

What can't go here?

The empty cells contain the possible values:

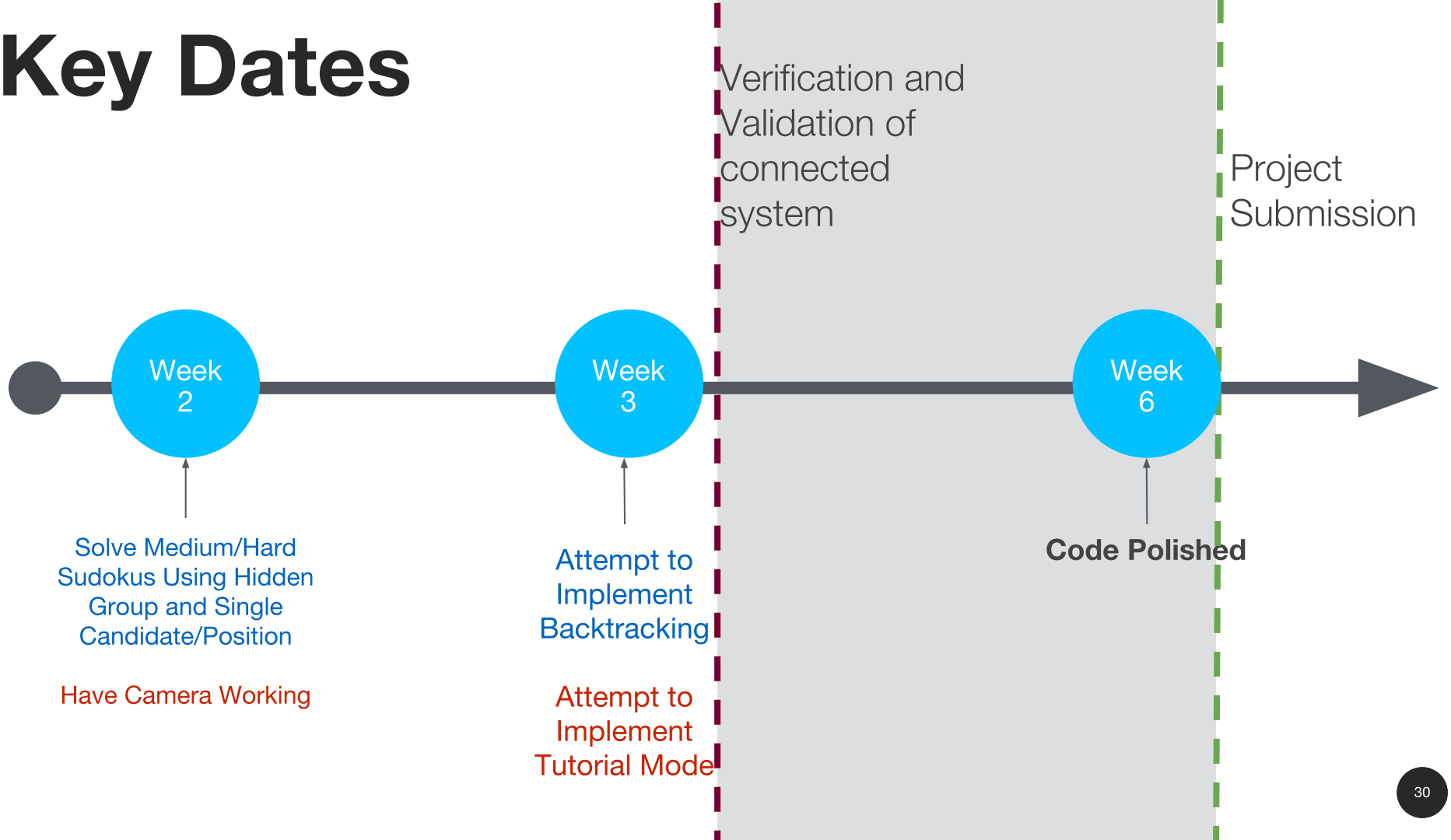
{4,9}
{4,6}
{2,4}
{1,3}
{2,6}
{1,2,3} -> {1,3}

We have only 2 places where 1 and 3 can go, so it is impossible for 2 to be in the last cell

The digital algorithm very cool! The Q&A is coming up *wink* *wink*

Project Timeline

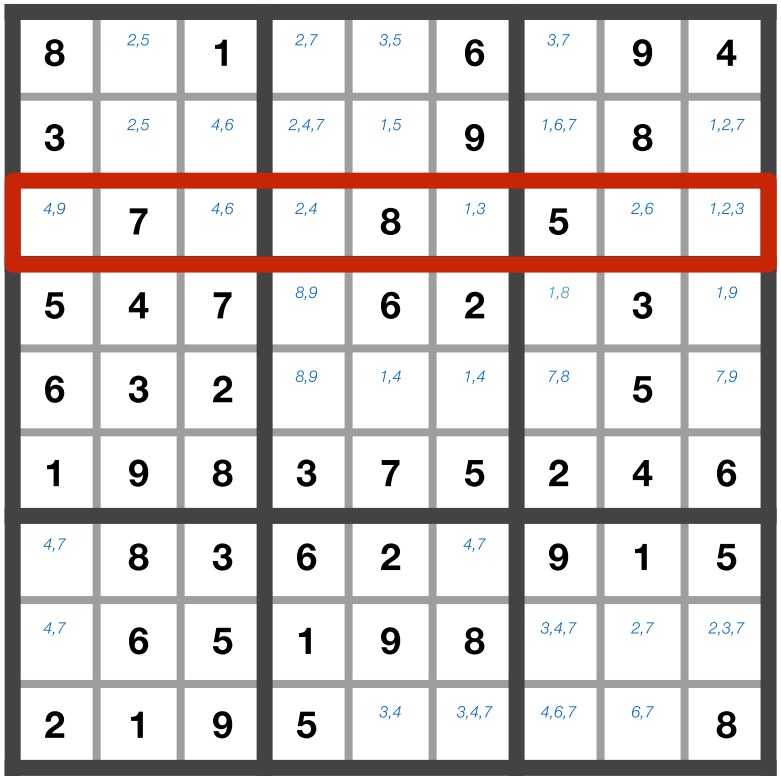
Key Dates


Verification and Validation of connected system
Project Submission
Week 2
Week 3
Week 6
Solve Medium/Hard Sudokus Using Hidden Group and Single Candidate/Position
Have Camera Working
Attempt to Implement Backtracking
Attempt to Implement Tutorial Mode
Code Polished

Thank you for your attention

11.06.2018

Magson Gao, Shreeyam Kacker



What can't go here?

The empty cells contain the possible values: 987654321

{4,9} = 100001000

{4,6} = 000101000

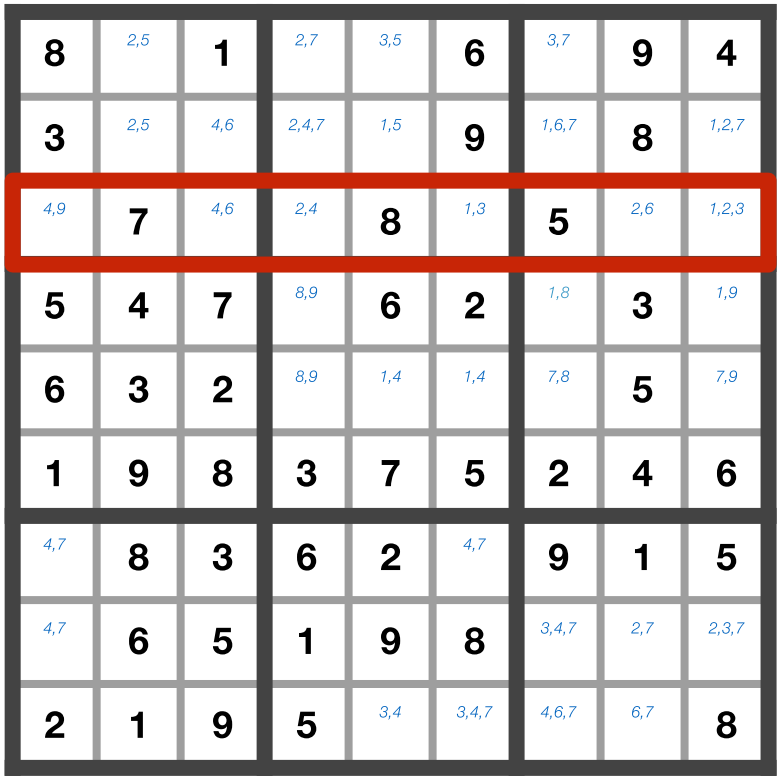
{2,4} = 000001010

{1,3} = 000000101

{2,6} = 000100010

{1,2,3} = 000000111

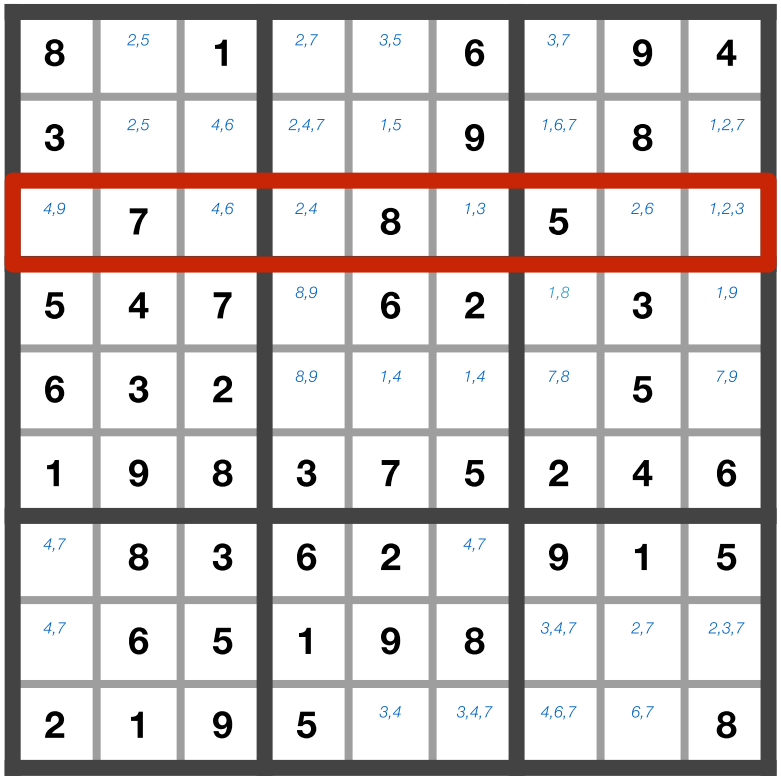
In this algorithm we iterate over each set



What can't go here?

Steps*

- ➔ Add up the number of 1s of the mask we are working on (let's call the number of 1s ***X*** and the mask we are working on ***Y***).
- ➔ AND mask ***Y*** with every other possibilities mask (this includes the ones that are solved, though **these always give 0**). Let us call the results of the AND operations ***A***.
- ➔ If the number of 0s in ***A*** are equal to (`9-X`) then a **hidden group** is detected
- ➔ If the number of masks in ***A*** with (# of 1s = ***X***) is equal to (`X-1`) a **naked group** exists between the ***Y*** and the masks in these positions.
- ➔ For every mask in the **hidden group**, replace it with the mask stored in ***A***. For every mask outside the **naked group** replace the mask with (`mask & ~Y`).



What can't go here?

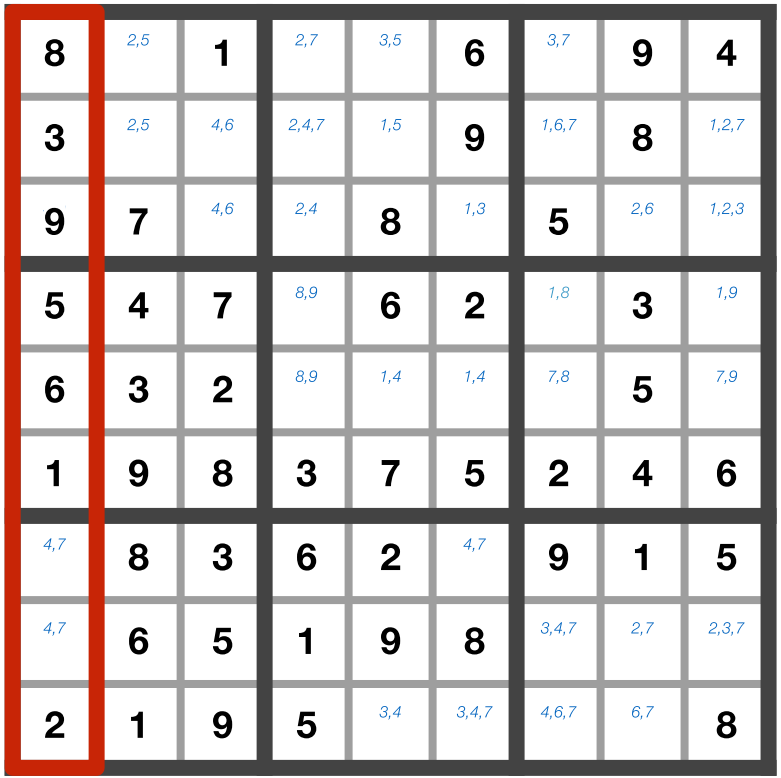
The empty cells contain the possible values:

```
                987654321   A:987654321
{4,9}   =       100001000   ->000000000
{4,6}   =       000101000   ->000000000
{2,4}   =       000001010   ->000000000
{1,3}   =       000000101   = Y
{2,6}   =       000100010   ->000000000
{1,2,3} =       000000111   ->000000101
X = 2,
Number of masks = 0 in A = 4 + 3
Number of masks with   2×1s = 1
```

*This method works for all hidden pairs and the vast majority of other worthwhile group sizes

Naked Group Setup

- ➔ Suppose we have **N** numbers which are the **only** possible values for **N** positions
- ➔ Then **all N positions must contain one of the N numbers. No other position in the row/column/square can contain the N numbers.**



What can't go here?

The empty cells contain the possible values:

```
987654321 A: 987654321
{4,9} =    100001000 -> 000001000
{4,7} =    001001000 =  Y
{4,7} =    001001000 -> 001001000
X = 2,~Y =110110111
Number of masks = 0 in A = 0 + 6
Number of masks with   2×1s = 1
```

The 3rd cell must be a 9