CS534 Spring 2019 Project Part Ⅱ --Sudoku

Presentation by: Yichen Li, Jiayi Li

Introduction

What is Sudoku?

Sudoku is a logic-based, combinatorial number-placement puzzle. The objective is to fill a 9×9 grid with digits so that each column, each row, and each of the nine 3×3 subgrids that compose the grid contain all of the digits from 1 to 9. The puzzle setter provides a partially completed grid, which for a well-posed puzzle has a single solution.

Introduction

What we do?

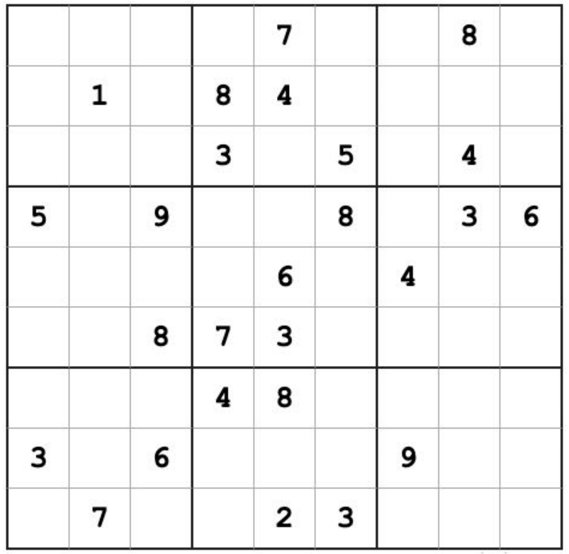
We implement a program that could solve classic 9*9 sudoku problem. Beside that, we expand our program and make it compatible with all n*n sudoku problem.

Detailed description

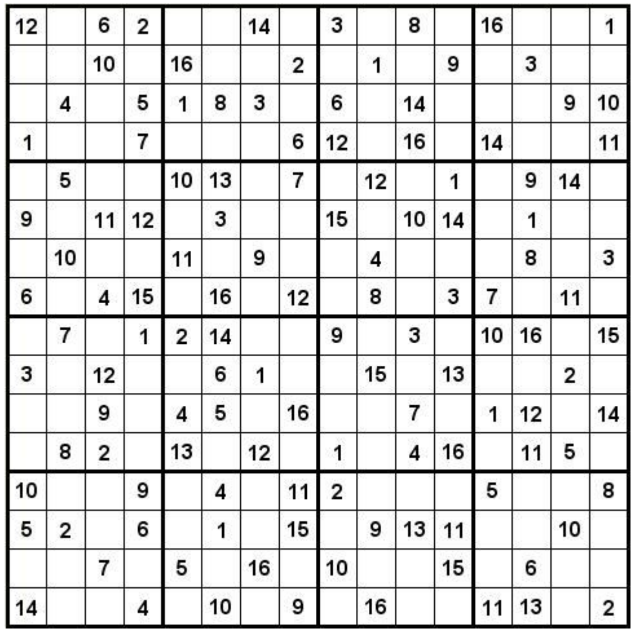
Variables: [0,0] [1,0] [2,3]........

Values: 0,1,2,3,...,dimension (dimension=9/16/25/...)

Constraints: Alldiff (row), Alldiff(collum), Alldiff(3*3 / 4*4 / 5*5 /…)



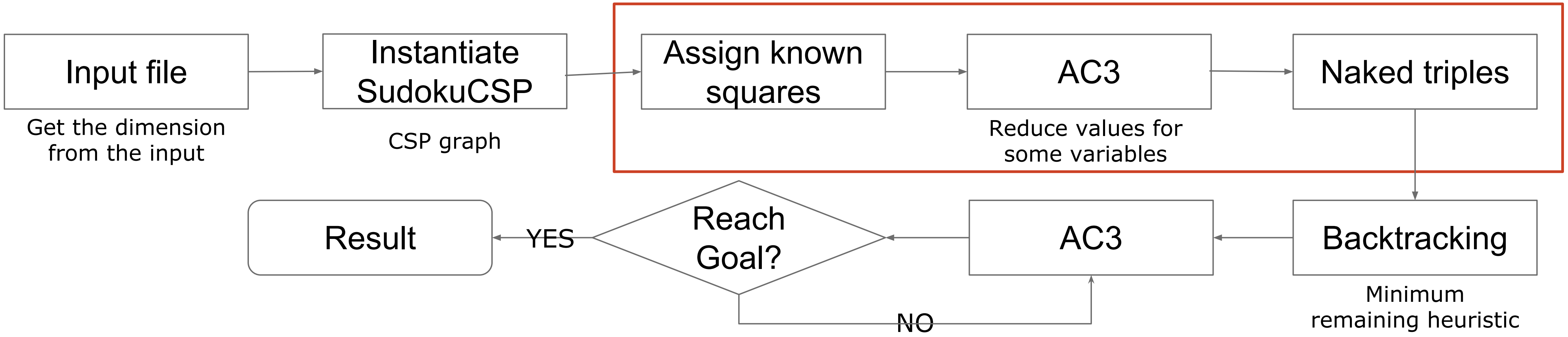
				7			8	
	1		8	4				
			3		5		4	
5		9			8		3	6
				6		4		
		8	7	3				
			4	8				
3		6				9		
	7			2	3			



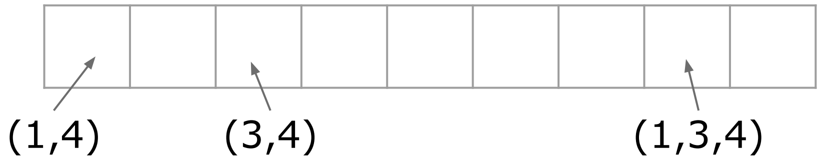
12		6	2			14		3		8		16			1
		10		16			2		1		9		3		
	4		5	1	8	3		6		14				9	10
1			7				6	12		16		14			11
	5			10	13		7		12		1		9	14	
9		11	12		3			15		10	14		1		
	10			11		9			4				8		3
6		4	15		16		12		8		3	7		11	
	7		1	2	14			9		3		10	16		15
3		12			6	1			15		13			2	
		9		4	5		16			7		1	12		14
	8	2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

Implementation

- Class csp(variable, domain, neighbours, constraints)
  - Extends class of part 1



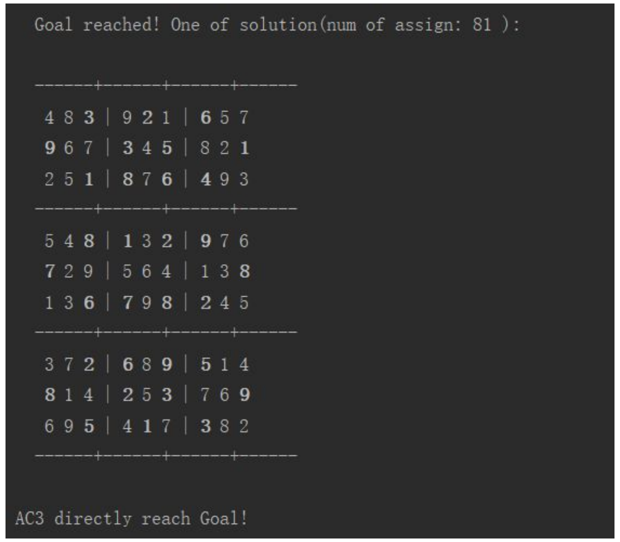
- Naked triples
  - In any unit (row, column or box), find three squares that each have a domain that contains the same three numbers or a subset of those numbers



Test Result

9*9

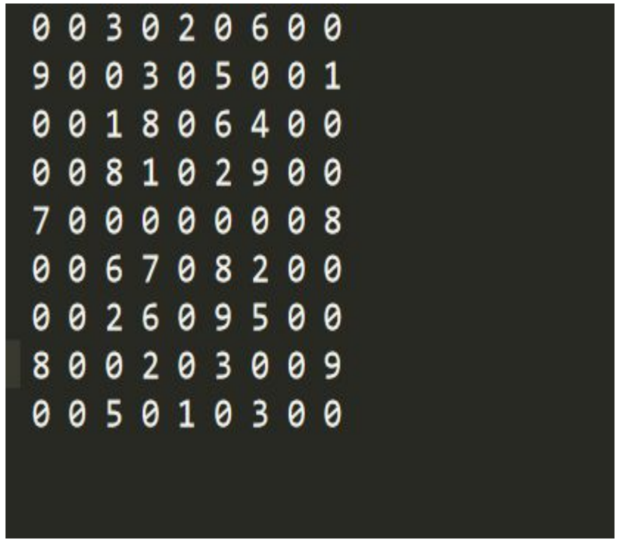
Directly solved by AC-3



```
 Goal reached! One of solution(num of assign: 81 ):

 ------+------+------+------
  4 8 3 | 9 2 1 | 6 5 7
  9 6 7 | 3 4 5 | 8 2 1
  2 5 1 | 8 7 6 | 4 9 3
 ------+------+------+------
  5 4 8 | 1 3 2 | 9 7 6
  7 2 9 | 5 6 4 | 1 3 8
  1 3 6 | 7 9 8 | 2 4 5
 ------+------+------+------
  3 7 2 | 6 8 9 | 5 1 4
  8 1 4 | 2 5 3 | 7 6 9
  6 9 5 | 4 1 7 | 3 8 2
 ------+------+------+------

AC3 directly reach Goal!
```

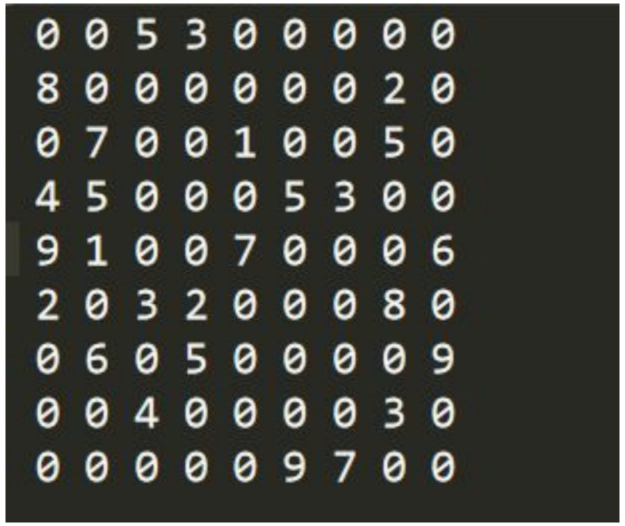


```
0 0 3 0 2 0 6 0 0
9 0 0 3 0 5 0 0 1
0 0 1 8 0 6 4 0 0
0 0 8 1 0 2 9 0 0
7 0 0 0 0 0 0 0 8
0 0 6 7 0 8 2 0 0
0 0 2 6 0 9 5 0 0
8 0 0 2 0 3 0 0 9
0 0 5 0 1 0 3 0 0
```

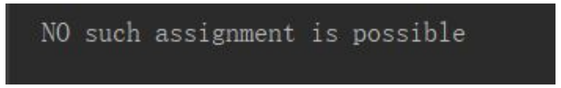
Test Result

9*9

No result



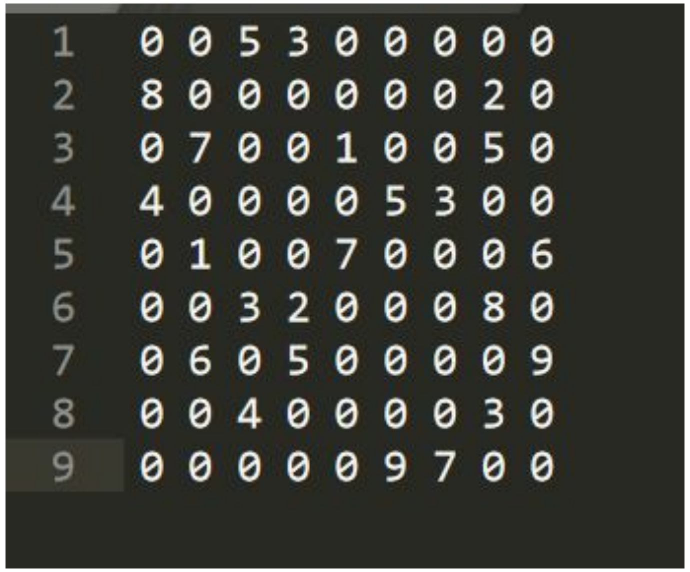
```
0 0 5 3 0 0 0 0 0
8 0 0 0 0 0 0 2 0
0 7 0 0 1 0 0 5 0
4 5 0 0 0 5 3 0 0
9 1 0 0 7 0 0 0 6
2 0 3 2 0 0 0 8 0
0 6 0 5 0 0 0 0 9
0 0 4 0 0 0 0 3 0
0 0 0 0 0 9 7 0 0
```



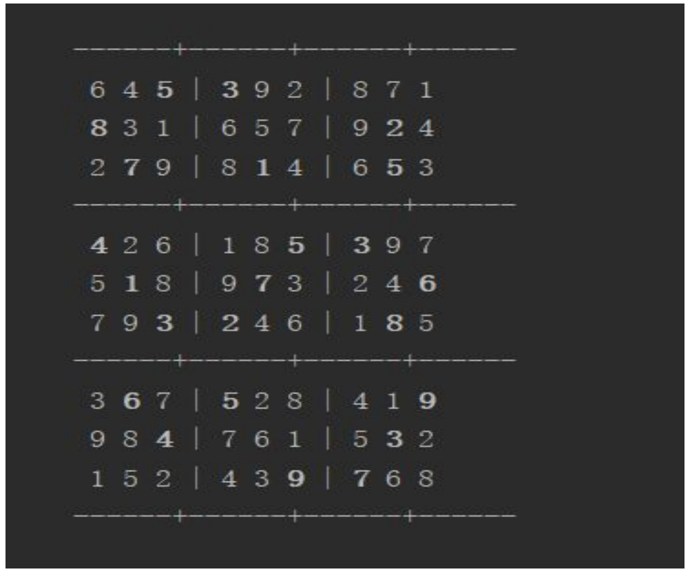
```
NO such assignment is possible
```

Test Result

9*9



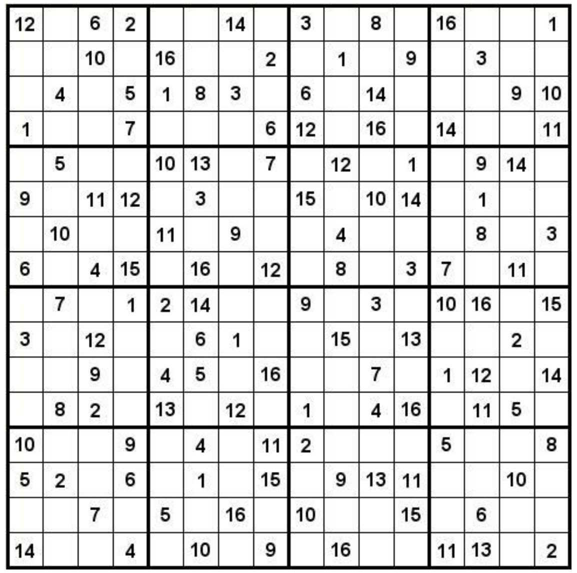
```
0 0 5 3 0 0 0 0 0
8 0 0 0 0 0 0 2 0
0 7 0 0 1 0 0 5 0
4 0 0 0 0 5 3 0 0
0 1 0 0 7 0 0 0 6
0 0 3 2 0 0 0 8 0
0 6 0 5 0 0 0 0 9
0 0 4 0 0 0 0 3 0
0 0 0 0 0 9 7 0 0
```



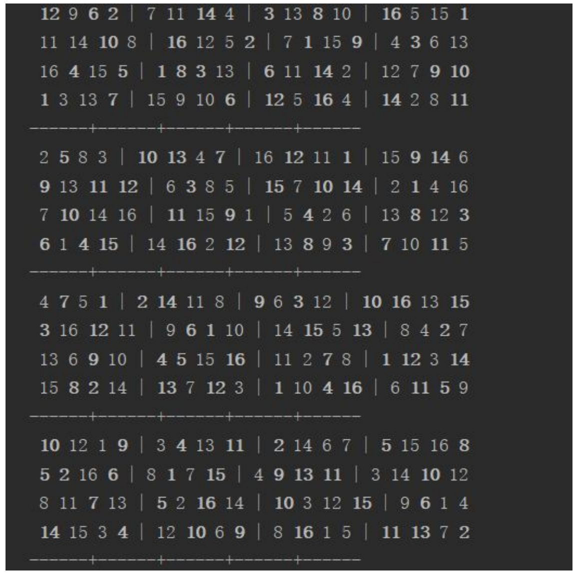
```
------+-------+-------+------
 6 4 5 | 3 9 2 | 8 7 1
 8 3 1 | 6 5 7 | 9 2 4
 2 7 9 | 8 1 4 | 6 5 3
------+-------+-------+------
 4 2 6 | 1 8 5 | 3 9 7
 5 1 8 | 9 7 3 | 2 4 6
 7 9 3 | 2 4 6 | 1 8 5
------+-------+-------+------
 3 6 7 | 5 2 8 | 4 1 9
 9 8 4 | 7 6 1 | 5 3 2
 1 5 2 | 4 3 9 | 7 6 8
------+-------+-------+------
```

Test Result

16*16



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



```
 12 9 6 2 | 7 11 14 4 | 3 13 8 10 | 16 5 15 1
 11 14 10 8 | 16 12 5 2 | 7 1 15 9 | 4 3 6 13
 16 4 15 5 | 1 8 3 13 | 6 11 14 2 | 12 7 9 10
 1 3 13 7 | 15 9 10 6 | 12 5 16 4 | 14 2 8 11
--------+--------+--------+--------+--------
 2 5 8 3 | 10 13 4 7 | 16 12 11 1 | 15 9 14 6
 9 13 11 12 | 6 3 8 5 | 15 7 10 14 | 2 1 4 16
 7 10 14 16 | 11 15 9 1 | 5 4 2 6 | 13 8 12 3
 6 1 4 15 | 14 16 2 12 | 13 8 9 3 | 7 10 11 5
--------+--------+--------+--------+--------
 4 7 5 1 | 2 14 11 8 | 9 6 3 12 | 10 16 13 15
 3 16 12 11 | 9 6 1 10 | 14 15 5 13 | 8 4 2 7
 13 6 9 10 | 4 5 15 16 | 11 2 7 8 | 1 12 3 14
 15 8 2 14 | 13 7 12 3 | 1 10 4 16 | 6 11 5 9
--------+--------+--------+--------+--------
 10 12 1 9 | 3 4 13 11 | 2 14 6 7 | 5 15 16 8
 5 2 16 6 | 8 1 7 15 | 4 9 13 11 | 3 14 10 12
 8 11 7 13 | 5 2 16 14 | 10 3 12 15 | 9 6 1 4
 14 15 3 4 | 12 10 6 9 | 8 16 1 5 | 11 13 7 2
--------+--------+--------+--------+--------
```

Conclusion

- Good expansibility
  - Theoretically support all n * n sudoku problem
- Good efficiency when solve classical sudoku
  - When solving 9*9 and 16*16 sudoku problem, our program has a good efficiency.
- Low efficiency when solve complicated sudoku problem
  - When solving 25*25 sudoku, our program spend a lot of time and space.

Thank you!

Questions

Backup Slides