Artificial Intelligence

COMS 4701 – Fall 2014

http://www1.ccls.columbia.edu/~ansaf/4701/

Home Work n°4: Constraint Satisfaction Problems

Due Saturday December 6th, 2014 @ 11:55pm

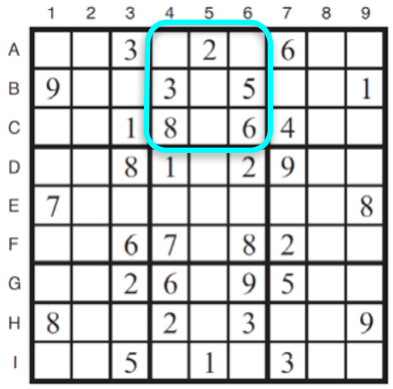
Problem 1: CSP for Sudoku Puzzle

The focus of this homework is constraint satisfaction problems. You will be implementing AC-3 and backtracking to solve Sudoku puzzles.

The rule of Sudoku puzzle is just to fill a 9*9 grid with digits so that each column, each row and each of the nine 3*3 sub-grids (also called "boxes") contains all of the digits 1 to 9.

Questions

Consider the Sudoku puzzle as pictured below. Each variable is named by its row and its column. Each variable must be assigned a value from 1 to 9 subject to the constraint that no two cells in the same row/column/box may contain the same value.



	1	2	3	4	5	6	7	8	9
A			3		2		6		
B	9			3		5			1
C			1	8		6	4		
D			8	1		2	9		
E	7								8
F			6	7		8	2		
G			2	6		9	5		
H	8			2		3			9
I			5		1		3		

1. (5 points) List the variables in the upper middle box in the blue circle and their corresponding initial domains.
2. (5 points) Reduce the domain for the four unassigned variables in question 1 by enforcing the arc constraints with the entire puzzle. List their new domains.
3. (5 points) Assume we have to choose one of the four unassigned variables in question 2 to explore further. Using minimum remaining value heuristic, which variable or variables should we explore next?
4. (5 point) Assume we choose A4 to explore next and assume that A6, B5, C5 are the last three remaining variables waiting to be assigned. Using least constraining value rule, which value of A4 should be tried first?
5. (30 points) Implement the AC-3 algorithm. Test your code on the provided set of puzzles *sudokus.txt*. Report the number of puzzles you can solve.
6. (50 points) Implement backtracking using minimum remaining value heuristic. Order of values to be tried for each variable is up to you. When a variable is assigned, apply forward checking to reduce variables' domains. Test your code on the provided set of puzzles *sudokus.txt*. Print your solution to each sudoku after solving them and save the output in a file called *output.txt*.

Files provided

hw4_UNI.py - Template Python file
sudokus.txt - Input sudoku set

hw4_UNI.py

In the homework, we use the variable names “Al” through “A9” for the top row (left to right), down to “I1” through “I9” for the bottom row, therefore each sudoku is represented as a dictionary variable named *sudoku* mapping from cell name which is a string to cell value which is an int. For example, *sudoku*[“A1”] = 1.

sudokus.txt

It contains a set of sudoku puzzles. Each line has 81 digits and is an individual sudoku puzzle. The first 8 digits are the first row of the sudoku and the second 8 digits are the second row and so on. If a digit is 0, it means that the cell has no assigned value. For example, the first line in *sudokus.txt* is

003020600900305001001806400008102900700000008006708200002609500800203009005010300

which represents the sudoku puzzle shown in the first page.

Files to submit

hw4_UNI.py - Completed Python code
output.txt - Output of sudoku results
hw4_written_UNI.pdf - PDF of answers to questions 1 to 5