Artificial Intelligence
COMS 4701 – Summer 2016

Home Work n°4: Constraint Satisfaction Problems
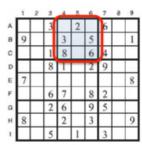### Due Friday July 1, 2016 @11:55pm
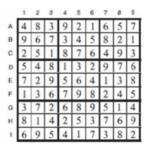

CSP for Sudoku Puzzle

The focus of this homework is constraint satisfaction problems. You will be implementing AC-3 and back-tracking 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. Implement the AC-3 algorithm. Test your code on the provided set of puzzles sudokus.txt. Report the number of puzzles you can solve.

2. (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_UNI.txt. Don't forget to replace "UNI" by your personal UNI.

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

        003020600900305001001806400008102900700000008006708200026095008002030090005010300
which represents the sudoku puzzle shown in the first page.

Files to submit

hw4_UNI.py   - Completed Python code

output_UNI.txt   - Output of sudoku results

hw4_written_UNI.pdf   - PDF explaining how to run your program, what you have done, you results and observations (including the number of sudoku you could solve and which one were solved), running time, and all relevant information.