CS3243: Introduction to Artificial Intelligence **(Due: October 18, 2019; 2359hrs)**

Assignment 2: Sudoku

Important: Read all the instructions below carefully before you start working on the assignment, and before you make a submission.

- You must do this assignment in **groups of two only**. Please write the names and matriculation numbers of your team members in the report and the code.
- Sign up your group on LumiNUS: Class and Groups → Class Groups
  - This time, you can form groups with anyone in the module.
  - i.e., there is no tutorial restriction on group forming.
  - Group size is strictly 2 students (No groups of 3 or individuals submissions are allowed. Reason: It helps us to manage logistics and grading).
- All sources of material must be cited. The University Academic Code of Conduct will be strictly enforced.

- **Submission instructions** (*These instructions are different from Assignment 1, please pay attention*)
- You need to submit two files on LumiNUS, your python code file and the report PDF.
  - Make only one submission (i.e., one set of two files) per group.
- File names:
  The code filename should be `sudoku_A2_xx.py`
  The report filename should be `sudoku_A2_xx.pdf`
  E.g. `sudoku_A2_03.py`, `sudoku_A2_03.pdf` (note that it is 03 and not 3).
- Points will be deducted for not following the naming convention. Please follow the file naming system closely.

- Kindly check that the submitted code will be able to run on SoC's Sunfire account. We will be using Sunfire (our UNIX server) to evaluate all submissions.

1 Background

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. Figure 1 shows a sample Sudoku puzzle[1].

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

Figure 1: A sample Sudoku puzzle

[1]This puzzle was created by Finnish mathematician Arto Inkala in 2012 and it was claimed to be the"hardest".

2 Problem Statement

Implement an efficient solver to solve the given Sudoku puzzle.
Here are the conditions/minor details you need to take note of/adhere to.

- You need to model the Sudoku puzzle as a CSP.
- You can explore any CSP algorithm/heuristic in this assignment and you are not restricted to the ones mentioned in the textbook. (Rationale: To give you an opportunity to explore things outside the textbook.)
  - We will time your solution and use the timing information as one of the (small) components for grading this assignment.
  - It may be advantageous to find a good algorithm/heuristic to speed up your solution.
  - However, please don't stress out too much about this! (Reason: Timing alone doesn't determine the full score for this assignment as such!)
  - If you have tried a few different algorithms/heuristics, please submit the solution that is your best. You can write about the other methods you tried in the report, and compare them with your submitted solution.
  - For the purpose of comparison and reporting, you can use the sample puzzle given in Figure 1.
- If you wish, you may compare the CSP solution you implemented with other techniques that can solve the Sodoku puzzle (eg. Dancing Links); however, this is plainly for your learning and will not be graded. For the purpose of this assignment, we only accept solutions that model Sudoku as CSP.

3 Logistics - Input/Output

3.1 Input

The input will be provided in a text file containing 9 lines to represent the Sudoku puzzle. Each line contains 9 numbers, which can be 0–9, where 0 represents an empty grid.
The given Sudoku puzzle is always:

1. 9×9 size
2. **valid** (i.e. each **column**, each **row**, and each of the nine 3×3 **subgrids** will not have duplicate non-zero numbers)
3. solvable, and the puzzle is well-formed (i.e., has exactly one solution).

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

Figure 2: Answer for the Figure 1 puzzle.

For example, the Sudoku Puzzle in Figure 1 is encoded in the text file as follows:

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

3.2 Output

The output should have the same format of the input. It should be **valid**, and only contain numbers 1–9 (i.e., there should be no blank space!).
For example, for the input in Figure 1, the only possible solution is shown in Figure 2, which should be encoded in the text file as follows:

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

4 Submission

Your submissions should contain the following: ONE python code file and ONE report in PDF. You will lose 5 points for not submitting either the report or the code file.

4.1 Code

Please use Python 2.6 (the default Python version on SoC's Sunfire) to do this assignment. The template has been provided to you (`sudoku_A2_xx.py`). You may import Python Standard libraries if you need (but no external libraries are allowed). Things to note:

- You may create new classes or functions if you think they are helpful (but all classes/functions should be within the same file). You SHOULD NOT modify the `main` function.
- Your answer should be returned by the `solve()` function; the `main` saves your answer in the correct format we expect. Hence, don't change the `main` function.
- **Executable**: If your program can not be executed, you will get 0 for your code components.
- **Correctness**: Please make sure your algorithm logic is correct. You will lose mark significantly if you can not pass our test cases.
- **Timing**: We will time your solution and use the timing information as one of the components in grading. Do note that your solution may take much longer to solve a **hard** puzzle than an **easy** one (and this is perfectly acceptable).

4.2 Report

Your report must be TYPE-WRITTEN (font size between 10-12pt). It should NOT be more than 2 pages. However, you may use additional pages for listing references.
Your report should contain at least these components:

- **Explanation**: Explain the algorithms and/or heuristics you use in your solution and why you choose it.
- **Analysis**: Discuss the optimization you have made in your implementation (if any). Discuss the time complexity of the algorithm that you implemented. You can also elaborate on other points that you think is valuable to discuss.