CSL452 : Artificial Intelligence

Lab-3

Group Members:

1. **Kartavya Ramnani (2013CSB1014)**
2. **Shinde Lav Chandrakant (2013CSB1032)**

Q2.) Formulate Sudoku as Boolean Satisfiability Problem.

This is the document describing the strategy I deployed while solving this problem and How i have coded this entire program. There is also a ReadMe file attached in the folder which can be used while running the code. As, no particular language was mentioned, I have used Python to code and Shell Script to structure the things.

1.) What is Sudoku ?

Sudoku is a 9X9 grid divided into nine 3X3 sub squares.The goal is to put numbers 1 to 9 in the grid so that each line, each column and each 3x3 square contains one and only one time each of the numbers.

Following is an example :

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

2.) What is Boolean Satisfiability Problem ?

A boolean satisfiability problem includes number of propositional boolean variables and we have to find a satisfying assignment such that the final result comes out to be True. A Conjunctive Normal Form (CNF) is a form which includes conjunction of clauses. A clause is a disjunction of boolean variables. So, the idea is to convert the Sudoku problem into CNF and feed it to a SAT solver which will provide a satisfying assignment which can then be used to reach to the solution of Sudoku.

3.) The Propositional Variables :

For each cell in the sudoku, i have dedicated 9 propositional variables for each value from 1 to 9 a cell can have. This means, the variable is X_{ijk} where i and j give the row and column number of the cell respectively and k gives the value it holds (from 1 to 9). Therefore, there will be 9 variables for each cell. As there are 81 cell, this implies there are **729** propositional variables.

4.) The Cell Constraint :

Now, each cell must have exactly one value from 1 to 9.

So, the following clauses were added :

$X_{111}\ X_{112}\ X_{113}\ \ldots.\ X_{119}$

The above clause will imply that the [1,1] cell will have at least one value from 1 to 9. Then we need to solve the constraint that a cell has to have exactly one value.

5.) The Row Constraint :

Clauses consist of : Every element in a row having 1, Every element in a row having 2,.... Till 9 and no 2 elements in a row having same element.

6.) The Column Constraint :

Same as above but for the column.

7.) The Block Constraint :

Same as above but for the blocks.

These will generate the clauses for any general empty sudoku which will follow each constraint. For a sudoku, where some spaces have already been filled and then we have to add the filled clauses which will help in finding the satisfying assignment.

8.) The Filled Constraint :

Suppose, if [1,1] has the value 6, then X_{116} will be true. We repeat this for every given value and put the corresponding propositional variable to be true.

This will give around 12000 clauses which can be converted to DIMACS form and feed to Minisat for finding the satisfying assignment. Normally, Minisat assigns False to every variable and then start guessing to find the satisfying assignment. This is extremely fast and solves one sudoku in ~ 0.01 second. The final output is stored in the finalanswer.txt file which is the desired output.

How To Run the Code !!

Some Requirements assumed are :

1. Sudoku has to be 9X9 only.
2. Any number of instances of Sudoku has to be entered in the “p.txt” file in the folder Problem2 as the name of the input file has not be asked and is assumed to be “p.txt”.
3. Minisat must be installed on the machine, I am running the command minisat on the command line itself using Shell Scripting.

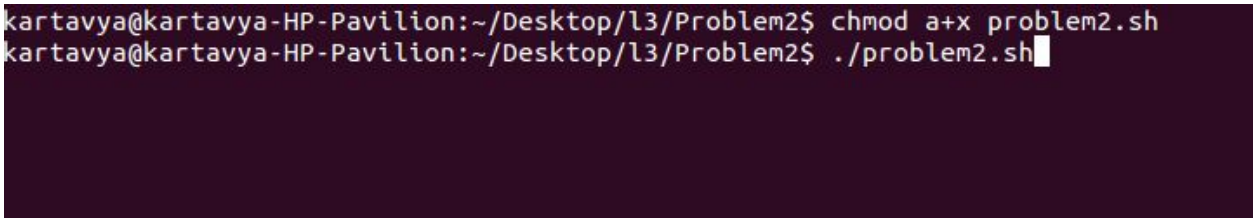
To Run The Code, Please do the following :

1.) Open the command line and enter the folder Problem2.

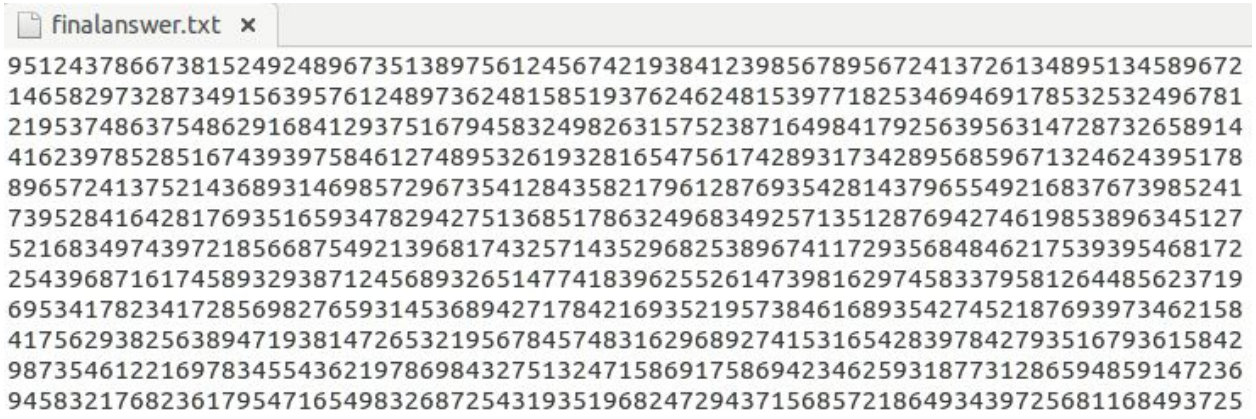
2.) Run the Script with the following commands :

   a.) Chmod a+x problem2.sh

   b.) ./problem2.sh



3.) The Final Output is stored in the desired form is “finalanswer.txt” file.



And the command line will look like (well, this multiplied by the number of sudoku instances) :