**SUDOKU SOLVER REPORT**

**NISHANT S  
MT2014079**

Sudoku solver is designed to solve sudoku puzzle of the order ranging from 2x2 to 10x10. The algorith runs in five different steps. The steps involved are:

1. Single Candidate

2. Single Square

3. Locked Set

4. Number Claim

5. Algorithm-X using Dancing Links

**Building Sudoku Board**

Sudoku Puzzle is stored in n²xn² structure array where n is the order of the sudoku puzzle. Each node of the stucture contains the following values: int value, Link possibles. Value will store the given puzzle, if blank it will be stored as zero. Possibles is ment to store the possible values as a linked list, that a particular position can hold.

Finding Possible Values

Once the build is over all the positions containing zero as value are taken one by one and curresponding possible values are found and entered in Link possibles. On basis of this possibility list the algorithm is going to fill the sudoku board.

**Single Candidate**

Single Candidate is based on the property that if a position can have only one value then it can be filled withe that value. The rule is implemented in following steps:

* Iterate through each entry in sudoku matrix.
* Choose the point where the number of entries in possibility list is exactly one.
* Fill the value with the one in possibility list.
* Delete the value in possibility.

**Single Square**

If within a region(i.e. a row, a column or a block) only one position can contain a value fill that postion with that value.

For implementing this we have to find that number. For that an array of integers of size n² is used where index of the array corresponds to values ranging from 1 to n². Then for each region the algorithm iterates through the possibility list and make a count for each value in the array. While updating count it keep track of corresponding row and column value too. Once iteration in one region is completed it checks the count and for whom ever the count is one the value is filled in corresponding row and column point.

**Locked Set**

A Locked Pair is 2 numbers that occur in 2 squares in the same row, column or nxn block. A set could also contain 2, 3, 4, numbers in 2, 3, 4 squares (etc). In that case we can remove those numberes in the set from possibilities of other squares in a region.

The algorithm involves finding squares having 'count' number of squares having 'count number of same same numbers as possibilities. Once this is found the values and the position of those squares are indexed. Later occurence of those values in possibilities of other squares whithin the region are removed. This step is repeated for count ranging from 2 to n². This is done for every column, row and box.

**Number Claim**

This Algorithm has four different varieties. The varieties can be named as:

1. Block to row

2. Block to column

3. Row to block

4. Column to block

i. Block to row and column

Consider a block with each square having varrying possibilities. Consider that among those possibilities one number comes in squares in a specific row in the block. Since a block should contain that number that number should defenitly come in one of the possitions inside that row within the block. Hence we can remove that number from possibility list of squares in the row outside the block. This can be similiarly happen in case of block to column too.

Algorithm will consider a row within a block and make set of numbers from the possibilities. From that set it removes the numbers which comes in possibilities of rest of the block. The final set of numbers are removed from possibilities of squares from rest of the row. The same is done for each row for each block. Similiarly this is done in case of columns too.

ii. Row and Column to Block

Consider a row which is part of a block. Lets say a number is only appearing in possibilities of squares whitin the part of the row which comes under the block. Since that number should be there in the row it should appear in one of the squares of the row within the block. Hence we can remove that number from possibilities of squares which comes under rest of the block. Same thing can be explained in case of column to block.

This is implemented in similiar way by taking set of numbers which comes under possibilities of squares of the row whic doesnt appeare in row outside block. Those numbers are deleted from possibilities of squares of rest of the block. Similiar implementation in case of column to block.

**Algorithm-X using dancing Links**

Most of the sudoku puzzles gets solved with use of the above algorithms. But in some case these are not enough. So this algorithm is used to solve such puzzles.

Sudoku problem can be considered as a special case of so called Exact Cover problem. Here Dr. Donald Knuth’s Dancing Links Algorithm is used to solve the sudoku puzzle which is used to solve an Exact Cover situation.

Exact Cover Problem

Exact Cover problem can be extended to a variety of applications that need to fill constraints. Sudoku is one such special case of the Exact Cover problem.

Given a matrix of 1 ’s and 0 ’s the Dancing Links will find a set or more of rows in which

exactly one 1 will appear for each column. For example, in Knuth’s paper figure 3 , a matrix

is given as:

0 0 1 0 1 1 0

1 0 0 1 0 0 1

0 1 1 0 0 1 0

1 0 0 1 0 0 0

0 1 0 0 0 0 1

0 0 0 1 1 0 1

Rows 1, 4, 5 are a set that solves this Exact Cover Puzzle.