Machine Problem 2: Sudoku Solving Algorithm using Backtracking on Sudoku Graph Color implemented Structure in C

Gimel David F. Velasco*, John Kenneth C. Fajardo
Department of Mathematics and Computer Science
University of the Philippines
gfvelasco@up.edu.ph

1. INTRODUCTION

Sudoku, originally called Number Place, is a logic-based combinatorial number placement puzzle. The puzzle consists of a 9x9 grid. The goal of sudoku is to assign digits to the empty cells so that every row, column and box contains exactly one instance of the digits from 1 to 9. The starting cells are assigned to constrain the puzzle such that there is only one way to finish it. A computer solver can make and unmake guesses fast by using different sudoku solving algorithms.

2. METHODOLOGY

The implemented sudoku solving algorithm in this program is called the 'Backtracking'. The backtracking algorithm implemented in this code considers the numbers 1 to 9 and checks if the number is suitable to be inserted into the cell. If not, the program will recurse and then consider another number. If all the numbers have been considered, the program will temporarily insert an unassigned value to the cell and then recurses to reconsider the preceding cell in was currently evaluating. The pseudocode for backtracking is as follows

Find row, col of an unassigned cell
If there is none, return true
For digits from 1 to 9
If there is no conflict for digit at row,col
assign digist to row,col and recursively try fill in the rest of the grid
if recursion is successful, return true
if not successful, remove digist and try another
end if
end for
If all digits have been tried and nothing worked, return false to trigger backtracking

This kind of cycle will go on and find another unassigned cell until there is no unassigned cell left. If all of this is done, the sudoku puzzle therefore is solved. But if all cases and numbers have been considered in every cell and the function already is finished, then there is no solution to the sudoku puzzle. The program uses boolean expressions True and False in which in this code is represented as '2' and '1' respectively.

2.1 Sudoku Graph Structure

The program or code is first implemented on arrays (Array implemented code is available upon request). After the array implemented backtracking sudoku solver is finished, the code then is manually converted so that a vertex-edge structure is implemented in the code. The program follows a sudoku graph wherein a single vertex is connected to 8 of its 'row mates' and 8 of its 'column mates' and since the box already is covered by the row and column, there will be the remaining 4 of its 'box mates'.

The structure of the node is described in below:

```
struct node:
int row, col
char color[10]
struct node *templink
struct node *rowmates[8]
struct node *colmates[8]
struct node *boxmates[4]
```

where row and col is its corresponding position in the board, color is the corresponding color of the vertex as defined below

0 - "Black", 1 - "Brown", 2 - "Red", 3 - "Orange",
4 - "Yellow", 5 - "Green", 6 - "Blue", 7 - "Violet",
8 - "Grey", 9 - "White"

The mapping of the numbers to colors implemented in this code is as that of how numbers are mapped to a corresponding color in capacitors. To further explain the structure, the templink is the link that connects all of the vertices in the sudoku board into a linked list. This is so that a much more easier and ergonomic traversal can be performed. The rowmates, colmates and boxmates in the structure are array of pointers connecting to their corresponding row, column and box. This structure is the basic graph structure for building the sudoku graph.