

Sudoku Solver

Submitted by :

Iftekhar Alqurashi

43700033

Fatemah mohammed

43754270

Afnan Alharthi

43701166

Shahad Khalid

43650557



Introduction

- What is AI ?
 - AI traditionally refers to an artificial creation of human-like intelligence that can learn, reason, plan, perceive, or process natural language.
 - AI is a rapidly advancing technology, made possible by the Internet, that may soon have significant impacts on our everyday lives
- Games and AI
 - Game AI is mostly focused on which actions an entity should take, based on the current conditions.
 - In each case it is a thing that needs to observe its surroundings, make decisions based on that, and act upon them.

Problem definition

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

Putting the right number in the right place is a problem in Sudoku that you need to think about so that the same number doesn't repeat in the column, row and box.



Algorithm used : “Backtracking”

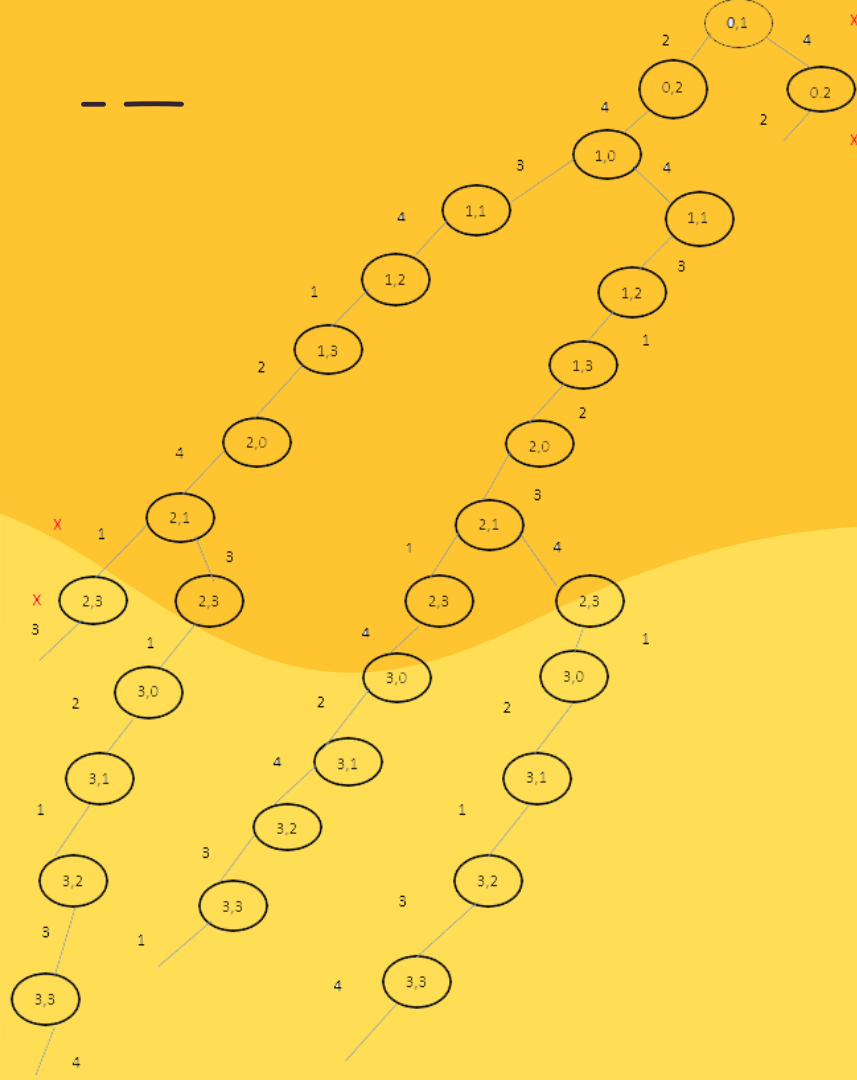
Backtracking search is an exhaustive search algorithm (depth-first search). It checks the row, column, and box and sees if the number is appropriate for this place. This process is repeated until the appropriate number is found.



Solve problem



Game tree



1			3
		2	

1	2	4	3
4	3	1	2
3	1	2	4
2	4	3	1

Code solution

Constructor to initialize class variables after getting information from the user

```
public Sudoku(int[][] board, int n)
```

we check if a possible number is already in a column

```
private boolean isInCol(int col, int number)
```

combined method to check if a number possible to a row,col position is ok

```
private boolean isOk(int row, int col, int number)
```

Display the board to the user

```
public void display()
```

```
import java.util.Scanner;

public class Sudoku {
    private int[][] board;
    public static final int EMPTY = 0; // empty cell
    public int SIZE;
    public int HALF_SIZE;

    public Sudoku(int[][] board, int n) {
        this.SIZE = n; // size of our sudoku grids
        this.HALF_SIZE = (int) Math.sqrt(SIZE);
        this.board = new int[SIZE][SIZE];

        for (int i = 0; i < SIZE; i++) {
            for (int j = 0; j < SIZE; j++) {
                this.board[i][j] = board[i][j];
            }
        }

        // we check if a possible number is already in a row
        private boolean isInRow(int row, int number) {
            for (int i = 0; i < SIZE; i++) {
                if (board[i][row] == number)
                    return true;
            }
            return false;
        }

        // we check if a possible number is already in a column
        private boolean isInCol(int col, int number) {
            for (int i = 0; i < SIZE; i++) {
                if (board[i][col] == number)
                    return true;
            }
            return false;
        }

        // we check if a possible number is in its 3x3 box
        private boolean isInBox(int row, int col, int number) {
            int r = row - row % HALF_SIZE;
            int c = col - col % HALF_SIZE;
            for (int i = r; i < r + HALF_SIZE; i++) {
                for (int j = c; j < c + HALF_SIZE; j++) {
                    if (board[i][j] == number)
                        return true;
                }
            }
            return false;
        }

        // combined method to check if a number possible to a row,col position is ok
        private boolean isOk(int row, int col, int number) {
            return !isInRow(row, number) && !isInCol(col, number) && !isInBox(row, col, number);
        }

        // Solve method, we will use a recursive Backtracking algorithm.
        // we will use helper methods in our solver.
        public boolean solve() {
            for (int row = 0; row < SIZE; row++) {
                for (int col = 0; col < SIZE; col++) {
                    // if this cell is empty, we will try to fill it
                    if (board[row][col] == EMPTY) {
                        // we try possible numbers from 1 to SIZE, number=1 to SIZE
                        for (int number = 1; number <= SIZE; number++) {
                            if (isOk(row, col, number)) {
                                // we put the number in the cell
                                board[row][col] = number;

                                // recursive sudoku constraints
                                if (solve()) { // we start backtracking recursively
                                    return true;
                                } else { // not a solution, we empty the cell and we continue
                                    board[row][col] = EMPTY;
                                }
                            }
                        }
                    }
                }
            }
            return false; // we return false
        }

        return true; // sudoku solved
    }

    public void display() {
        for (int i = 0; i < SIZE; i++) {
            for (int j = 0; j < SIZE; j++) {
                System.out.print(" " + board[i][j]);
            }
            System.out.println();
        }
        System.out.println();
    }
}
```

we check if a possible number is already in a row

```
private boolean isInRow(int row, int number)
```

we check if a possible number is in its inner box

```
private boolean isInBox(int row, int col, int number)
```

We will use a recursive Backtracking algorithm

```
public boolean solve()
```

Run code

The player consider 4x4 sudoku
as following:

```
Enter The Number Of rows in your sudouko
4
Enter Sudoku Data
1 0 0 3
0 0 0 0
0 0 2 0
0 0 0 0
Sudoku grid to solve
1 0 0 3
0 0 0 0
0 0 2 0
0 0 0 0

Sudoku Grid solved !
1 2 4 3
3 4 1 2
4 3 2 1
2 1 3 4
```

Conclusion

Sudoku is a strategy-based game to complete and fill with numbers the right way. In this project, we have provided a solution to the Sudoku problem. By following a backtracking strategy we avoid repeating the same number in a column, row, and box. Although we had trouble finding a solution, the backtracking strategy solved it. Backtracking checks the row, column, and box and sees if the number is appropriate for this place. This process is repeated until the appropriate number is found.



THANKS!

Does anyone have any questions?

References

References:

1. Sizer,B. The Total Beginner's Guide to Game AI. Gamedev. August 13, 2018. Availabe at: <https://www.gamedev.net/articles/programming/artificial-intelligence/the-total-beginners-guide-to-game-ai-r4942/>
2. Erickson,J. *Algorithms* (1st ed.). June 13, 2019
3. Saurel,S. Build a Sudoku Solver in Java.medium. June 2,2018 .Availabe at: <https://medium.com/javarevisited/build-a-sudoku-solver-in-java-part-1-c308bd511481>
4. Wikipedia contributors. Sudoku. Wikipedia, The Free Encyclopedia.November 24,19.Availabe at: <https://en.wikipedia.org/wiki/Sudoku>