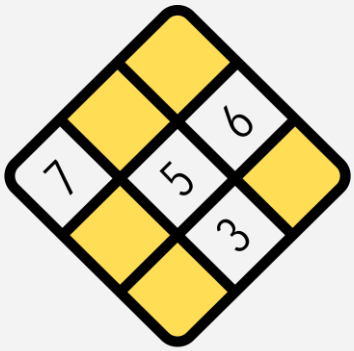
Nov 2019

College of Computers and Information Technology, Computer Science Department



Sudoku Solver

Submitted by:

Iftekhar Alqurashi	43700033
Fatemah mohammed	43754270
Afnan Alharthi	43701166
Shahad Khalid	43650557

Supervised by:
Dr.Haneen Al-Gethami

Abstract

The sudoku game falls under puzzle games.The simplicity of the puzzle’s structure and the low requirement of mathematical skills caused people to have an enormous interest in accepting challenges to solve the puzzle .Therefore, developers have tried to find algorithms in order to generate a variety of puzzles for human players so that they could be even solved by computer programming .In our project, we use the JAVA language programming with method Backtracking.

Backtracking is an algorithmic-technique for solving problems recursively by trying to build a solution incrementally, In sudoku solving Problems, we try filling numbers one by one.

if we find that the current number cannot lead to a solution, we remove it (backtrack or kill) and try the next number.

Table of contents:

Abstract 2

Table of figures 4

Introduction..... 5

Artificial Intelligence (AI) 5

Games and AI 5

Introduction to Relative Work 5

Problem definition..... 6

Solve the problem.....7

Game tree 8

Code solution..... 9

Run code..... 12

Conclusion..... 13

References 14

Table of figures:

Figure 1 6

Figure 2 7

Figure 3 7

Figure 4 8

Figure 5 12

Introduction

Artificial intelligence is a technology that is already impacting how users interact with, and are affected by the Internet.

Artificial Intelligence (AI) is a rapidly advancing technology, made possible by the Internet, that may soon have significant impacts on our everyday lives. AI traditionally refers to an artificial creation of human-like intelligence that can learn, reason, plan, perceive, or process natural language. These traits allow AI to bring immense socioeconomic opportunities, while also posing ethical and socio-economic challenges.

Games and AI

Game AI is mostly focused on which actions an entity should take, based on the current conditions. This is what the traditional AI literature refers to as controlling‘ intelligent agents’ where the agent is usually a character in the game – but could also be a vehicle, a robot, or occasionally something more abstract such as a whole group of entities , In each case it is a thing that needs to observe its surroundings, make decisions based on that, and act upon them.

Introduction to Relative Work

Sudoku is a puzzle game based on numbers and contains a grid of squares n × n, divided into rows and columns of each row and each column must contain the numbers from 1 to n but provided that it does not exist before in the row or column.
Sudoku is a brain game that is useful for brain training and stimulation.
Sudoku: Usually play in newspapers but we will develop them using Java programming language to become an AI application.
French newspapers featured variations of the Sudoku puzzles in the 19th century, and the puzzle has appeared since 1979 in puzzle books under the name Number Place.

Problem statement

Putting the right number in the right place is a problem in Sudoku that you need to think about so that the same number does not repeat in the column, row and box (see Figure1).

- In the row

Adding a number to the row so that the number added in the row is not repeated is a problem that needs to be resolved.

- In the column

The number to be added should not be repeated in the column is the same problem we faced in the row for example add a number 3 to the column It will be difficult to know if there is a number like him in the column.

- In the box

In the box we do the same thing the added number is not repeated in the box. Each box has a set of non-recurring numbers in the same box.

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

Figure 1

Solve the problem :

The algorithm used for implementing code is backtracking. Backtracking search is an exhaustive search algorithm (depth-first search), it tries to construct a solution to a computational problem incrementally, one small piece at a time. Whenever the algorithm needs to decide between multiple alternatives to the next component of the solution, it recursively evaluates every alternative and then chooses the best one.

Initial state :

1			3
		2	

Figure 2

Goal state :

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

Figure 3

Game tree :

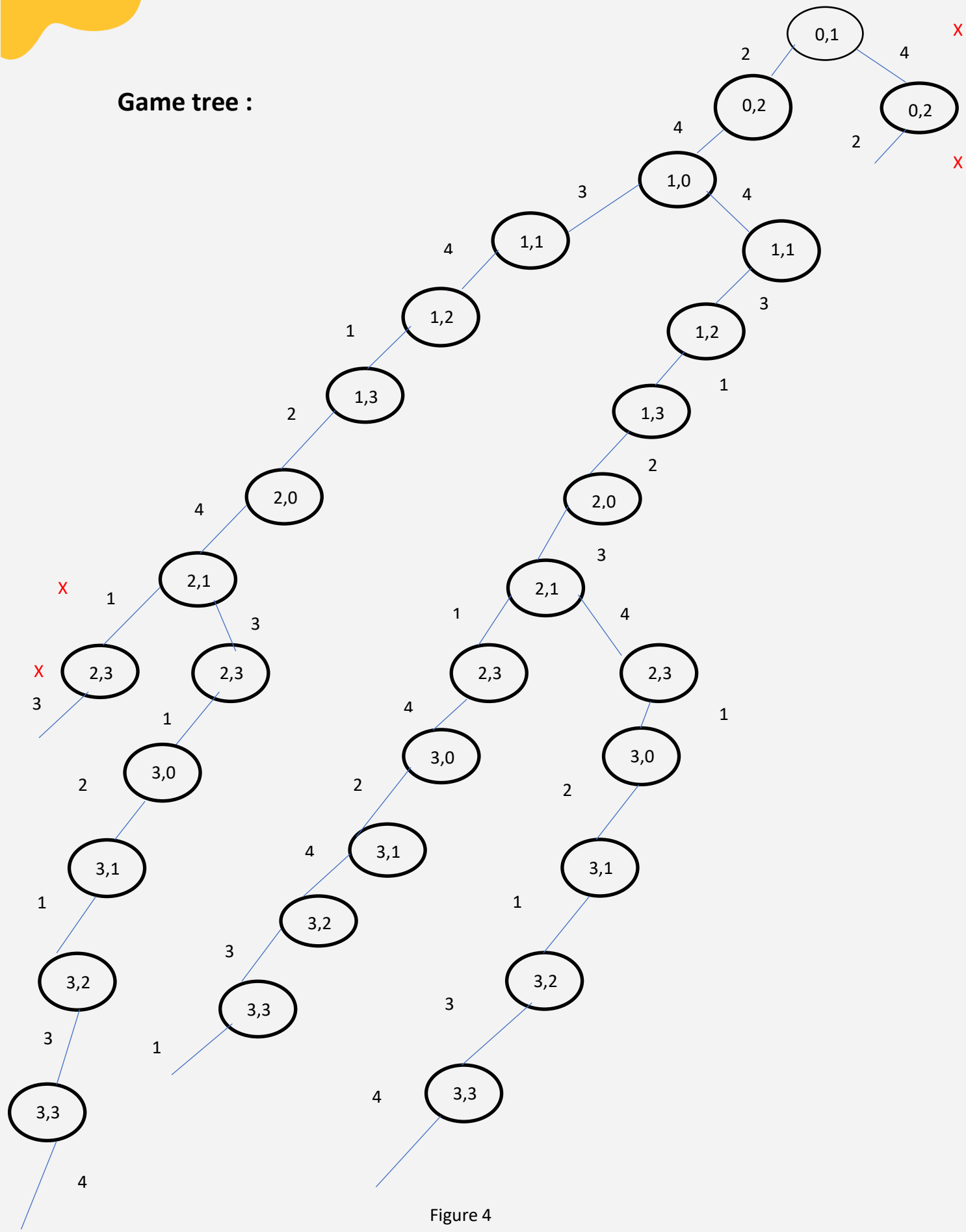


Figure 4

Code solution :

```
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[row][i] == 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 inner 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.

  public boolean solve () {
   for (int row = 0; row < SIZE; row++){
    for (int col = 0; col < SIZE; col++){
  //  we search an empty cell
     if (board[row][col] == EMPTY) {
 //  we try possible numbers
       for (int number = 1; number <= SIZE; number++) {
         if (isOk(row, col, number)) {
  // number ok. it respects sudoku constraints
           board[row][col] = number;

           if (solve()) { // we start backtracking recursively
             return true;
         } else { // if 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();
    }
```

```
 public static void main(String[] args) {
 Scanner scan = new Scanner(System.in);
System.out.println("Enter The Number Of rows in your sudouko");
 int rowNum = scan.nextInt();

//  defining 2D array to hold Sudoku data
 int[][] matrix = new int[rowNum][rowNum];
 System.out.println("Enter Matrix Data");

  for (int i = 0; i < rowNum; i++) {
     for (int j = 0; j < rowNum; j++) {

        matrix[i][j] = scan.nextInt();
     }
}
         Sudoku sudoku = new Sudoku(matrix,rowNum);

         System.out.println("Sudoku grid to solve");
         sudoku.display();

          //we try resolution
         if (sudoku.solve()) {
                 System.out.println("Sudoku Grid solved !");
                 sudoku.display();
         }  else {
                 System.out.println("Unsolvable");}}}
```

Run code :

```
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
```

Figure 5

Conclusion

Sudoku is a strategy-based game to complete and fill with numbers the right way. In this report, 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.

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