PROJECT REPORT

On

SUDUKO SOLVER

Bachelor of Technology
In
Computer Science & Engineering

Chandigarh Group of Colleges, Mohali

Submitted To:
Ms. Kamalinder Mam

Submitted By:
Vinay Kumar
1802899
CSE-V2

Department of CSE CEC,LANDRAN

ACKNOWLEDGEMENT

I am profoundly grateful to Ms. Kamalinder Mam for his expert guidance and continuous encouragement throughout to see that this project rights its target. I would like to express deepest appreciation towards Dr. Sukhpreet Kaur, HOD CSE CEC, Landran and Dr. Shelja Jhamb Chief CC Computer Science Department whose invaluable guidance supported me in this project. At last I must express my sincere heartfelt gratitude to all the staff members of Computer Engineering Department who helped us directly or indirectly during this course of work.

ABSTRACT

In our project we propose a method to solve the elements of a Sudoku Puzzle and providing a digital copy of the solution for it . The method involves a Web-based Sudoku solver. The solver is capable of solving a Sudoku directly from Website by just clicking on a button.

After applying appropriate algorithm to the acquired problem we use efficient area calculation techniques to recognize the enclosing box of the puzzle.

A container is created using HTML5 and CSS3 with different buttons to identify the digit positions. The actual solution is computed using a backtracking algorithm. Experiments conducted on various types of Sudoku questions demonstrate the efficiency and robustness of our proposed approaches in real-world scenarios.

The algorithm is found to be capable of handling cases of translation, perspective, time complexities , scaling, and background clutter.

Backtracking Algorithm is used to solve Sudoku puzzles problem. Backtracking is the process of trying solution, if its correct we proceed further else we backtrack and explore other solutions. Backtracking uses recursion to solve every problem which need backtracking to solve it.

Keywords- HTML5; Buttons; Div Container; recursive backtracking algorithm; time complexities; Web-based Sudoku solver; JavaScript,Recursion;Images;Texts;CSS selectors,Optimal Solution and Brute Force Approach.

Introduction

In real life we come across Sudoku puzzles of varying difficulty levels in newspapers and other text and digital media. It is a common leisure activity for a lot of people. However, it is observed that the solution is not always immediately available for the users verification. In most cases, people have to wait till the next day to check the solutions of the Sudoku they just solved. Hence our motivation for this project was to develop an application on an android device for this purpose. In our application, the user needs to capture a clean image of an unsolved Sudoku Puzzle, which then returns the complete solution of the same.

Figure 1.1: Example of Sudoku Puzzle

Sudoku is a logic based puzzle with the goal to complete a 9x9 grid so that each row, each column and each of the nine 3x3 boxes contain the numbers 1 through 9, given a partial flling to a unique solution. The problem itself is a popular brainteaser but can easily be used as an algorithmic problem, with similarities to the graph coloring problem only with fixed size and dependencies. The modern version of Sudoku was introduced in Japan in the eighties and has since then attracted a lot of programmers and therefore there are a great deal of different Sudoku solving algorithms available on the Internet, both implementations and construction methods.Sudoku solvers are interesting because they are practical applications of very theoretical problems, and most people are familiar with them..

Problem Statement & Objectives

The task is to solve the Sudoku puzzle in as much less time as possible.It can be done by investigating different techniques for solving Sudoku and comparing them for the most efficient solution. Sudoku itself can be solved using brute-force in a reasonable amount of time in most cases, but there are special cases where it takes a long time to brute-force. Therefore our task is to try to find efficient algorithms for all instances of the problem and evaluate them while using the optimal solver to get the solution for the Sudoku puzzle.

There are two main constraints that determine the efficiency of a optimal solver. They are Time consumption and Memory consumption whose degree of satisfaction determines the quality of a optimal Sudoku solver. These constraints usually vary from algorithm to algorithm.

2.1 Objectives

The objective of this project is to solve Sudoku with the use of recursive back tracking algorithm while satisfying the game constraints.

Constraints:

• Sudoku puzzle can only contain the numbers 1 through 9.

• A position constraint: Only 1 number can occupy a cell .

• A row constraint: Only 1 instance of a number can be in the row .

• A column constraint: Only 1 instance of a number can be in a column.

• A region constraint: Only 1 instance of a number can be in a region.

Sudoku Solver

Summary

A Smart Sudoku Solver is presented that can solve unsolved Sudoku with small amount of perspective. The algorithm can also give solution in cases of severe rotation such as when the Sudoku Puzzle is completely inverted. Our algorithm efficiently manages these problems. Since there doesn't exist any standard Sudoku data Set to test on, an own data Set from a Website was created and produced results on them.

We can set the Level of Difficulty of the Sudoku Solver Problem.There are three levels of difficulty which includes Easy ,Medium, Hard. However in general, for the testing were extracted from the web, work efficiently when algorithm is applied on it. These testing may contain problems which got solved and which not got solved.

Advantages

• A person can understand the logic behind the Sudoku Solver puzzle and try to solve Sudoku Puzzles anywhere he is needed to solve it.

• Back tracking algorithm takes less memory and time compared to Dancing links algorithm.

• A solution is guaranteed (as long as the puzzle is valid)

• Solving time is mostly unrelated to degree of difficulty

Disadvantages

•This approach may take more time in some of the cases and it can be optimized further for fluent working.

• It only contain some set of Sudoku Puzzles and doesn't provide variety of puzzles as it only contains 9x9 puzzles only.

Sudoku solving Algorithm

After the digits have been successfully recognized,the Sudoku puzzle can now be solved using a proper algorithm. We have used Recursive backtracking to solve the Sudoku puzzle.

Recursive backtracking

The algorithm to solve the Sudoku is based on a recursive backtracking strategy. We store the numbers obtained from the grid in the image, in a two dimensional array with the number 0 assigned to blank grid locations. To get a solution for the grid, we identify a blank location in the grid. Based on Sudoku rules, we identify a valid assignment by iterating through the numbers 1-9. We then try to recursively solve the grid with this new number placement. If there are no more grid locations which need to be filled, it is an indication that the grid is solved, and we return the solved grid back to the main algorithm.

On the other hand, if we fail to recursively find a solution for a grid, we backtrack and try a different number assignment for a location and try solving the grid again. If all combinations of number assignments are depleted without finding a solution, we conclude the grid is unsolvable and therefore the function will return false. For our purposes however, we assume that the image taken by the user is a valid Sudoku puzzle and our algorithm will always solve it irrespective of the difficulty level of the puzzle.

Algorithm

Find row, col of an unassigned cell If there is none, return true.
For digits from 1 to 9

a) If there is no conflict for digit at row,col assign digit to row,col and recursively try fill in rest of grid

b) If recursion successful, return true

c) Else, remove digit and try another If all digits have been tried and nothing worked, return false

TECHNOLOGY USED

- Operating System 64 Bit(Windows,Mac,Linux etc).

- **HTML5:** It is used to provide structure to the Website.Different buttons are created using HTML5 elements and images ,texts are also used to make it more understandable.

- **CSS3:** CSS3 is used to provide Website a cool look so Whenever someone come to website to play Sudoku solver the person not get bored. CSS3 various properties and selectors are used to make Website Interactive.

- **JavaScript:** JavaScript is used to write the logic behind the Sudoku Solver. The Algorithm is written in JavaScript which performs all operations. JavaScript Interact with HTML5 elements to perform different operations on the elements.

- **Visual Studio Code:** Visual Studio Code is the most used Code Editor for Web Development related work. It provides different Extensions to make work easier and auto complete option to make writing code in easier way.

- **Google Fonts:** Google fonts are used to provide interactive fonts to the text or Numbers which appear on the container containing different buttons.

- **Backtracking Algorithm :** Backtracking concept is used to solve Sudoku Puzzle. Backtracking is one of the most important concept of Data Structures and Algorithms. Backtracking Basically uses Recursion to solve all the problems.

Source Code With Working Images

HTML5

```
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Suduko Solver</title>
    <link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/css/bootstrap.min.css"
integrity="sha384-Gn5384xqQ1aoWXA+058RXPxPg6fy4IWvTNh0E263XmFcJlSA
wiGgFAW/dAiS6JXm" crossorigin="anonymous">

    <link rel="stylesheet" href="style.css">
    <script defer src="script.js"></script>
</head>

<body>
    <!-**********Navbar Section**********      -->
    <nav>
        <img class="starstyle" src="/Solution/Star.jpg">
        <div class="Sudsolver">
            <h1 style="font-size:60px">Suduko Solver</h1>
        </div>
        <img class="starstyle" src="/Solution/Star.jpg">
    </nav>
    <br><br>
<!-**********Sudoku Board Section**********     -->
    <div id="container">

        <div class="lsb tsb" id="0">

        </div>

        <div class="tsb ldb" id="1">

        </div>

        <div class="tsb ldb" id="2">

        </div>

        <div class="tsb lsb" id="3">

        </div>
        <div class="tsb ldb" id="4">
```

```
</div>

<div class="tsb ldb" id="5">

</div>
<div class="tsb lsb" id="6">

</div>

<div class="tsb ldb" id="7">

</div>
<div class="tsb rsb ldb" id="8">

</div>

<div class="lsb tdb" id="9">

</div>

<div class="ldb tdb" id="10">

</div>

<div class="ldb tdb" id="11">

</div>

<div class="lsb tdb" id="12">

</div>
<div class="ldb tdb" id="13">

</div>

<div class="ldb tdb" id="14">

</div>
<div class="lsb tdb" id="15">

</div>

<div class="ldb tdb" id="16">

</div>
<div class="rsb ldb tdb" id="17">

</div>
```

```
<div class="lsb bsb tdb" id="18">

</div>

<div class="bsb ldb tdb" id="19">

</div>

<div class="bsb ldb tdb" id="20">

</div>

<div class="bsb lsb tdb" id="21">

</div>
<div class="bsb ldb tdb" id="22">

</div>

<div class="bsb ldb tdb" id="23">

</div>
<div class="bsb lsb tdb" id="24">

</div>

<div class="bsb ldb tdb" id="25">

</div>
<div class="bsb rsb ldb tdb" id="26">

</div>

<div class="lsb" id="27">

</div>

<div class="ldb" id="28">

</div>

<div class="ldb" id="29">

</div>

<div class="lsb" id="30">

</div>
<div class="ldb" id="31">
```

```
</div>

<div class="ldb" id="32">

</div>
<div class="lsb" id="33">

</div>

<div class="ldb" id="34">

</div>
<div class="rsb ldb" id="35">

</div>

<div class="lsb tdb" id="36">

</div>

<div class="ldb tdb" id="37">

</div>

<div class="ldb tdb" id="38">
</div>

<div class="lsb tdb" id="39">

</div>
<div class="ldb tdb" id="40">

</div>

<div class="ldb tdb" id="41">

</div>
<div class="lsb tdb" id="42">

</div>

<div class="ldb tdb" id="43">

</div>
<div class="rsb ldb tdb" id="44">

</div>

<div class="lsb bsb tdb" id="45">
```

```
</div>

<div class="bsb ldb tdb" id="46">

</div>

<div class="bsb ldb tdb" id="47">

</div>

<div class="bsb lsb tdb" id="48">

</div>
<div class="bsb ldb tdb" id="49">

</div>

<div class="bsb ldb tdb" id="50">

</div>
<div class="bsb lsb tdb" id="51">

</div>

<div class="bsb ldb tdb" id="52">

</div>
<div class="bsb rsb ldb tdb" id="53">

</div>

<div class="lsb" id="54">

</div>

<div class="ldb" id="55">

</div>

<div class="ldb" id="56">

</div>

<div class="lsb" id="57">

</div>
<div class="ldb" id="58">

</div class="ldb">
```

```
<div class="ldb" id="59">

</div>
<div class="lsb" id="60">

</div>

<div class="ldb" id="61">

</div>
<div class="rsb ldb" id="62">

</div>

<div class="lsb tdb" id="63">

</div>

<div class="ldb tdb" id="64">

</div>

<div class="ldb tdb" id="65">

</div>

<div class="lsb tdb" id="66">

</div>
<div class="ldb tdb" id="67">

</div>

<div class="ldb tdb" id="68">

</div>
<div class="lsb tdb" id="69">

</div>

<div class="ldb tdb" id="70">

</div>
<div class="rsb ldb tdb" id="71">

</div>

<div class="lsb bsb tdb" id="72">

</div>
```

```
            <div class="bsb ldb tdb" id="73">

            </div>

            <div class="bsb ldb tdb" id="74">

            </div>

            <div class="bsb lsb tdb" id="75">

            </div>
            <div class="bsb ldb tdb" id="76">

            </div>

            <div class="bsb tdb ldb" id="77">

            </div>
            <div class="bsb lsb tdb" id="78">

            </div>

            <div class="bsb ldb tdb" id="79">

            </div>
            <div class="bsb rsb ldb tdb" id="80">

            </div>


        </div>
        <br>
        <!--***************Buttons Section********************-->
        <div id="buttons">
            <button type="button" class="btn btn-danger" id="generate-sudoku">Get
New
                Puzzle</button>    
            <button type="button" class="btn btn-success" id="solve">Solve</button>
        </div>
</body>

</html>
```

CSS3

```
body {
	background-color: black;
}


/*************************Navbar Section**********************/

nav {
	display: flex;
	align-items: center;
	justify-content: center;
}

.Sudsolver {
	background-color: black;
	color: white;
	height: 100px;
	text-align: center;
}

h1 {
	padding: 20px;
	font-weight: bolder;
	text-shadow: 2px 2px hotpink;
	font-family: Georgia, 'Times New Roman', Times, serif;
}

.starstyle {
	height: 100px;
}

h1:hover {
	text-shadow: 2px 2px #ff0000;
	font-size: 70px;
	font-weight: bolder;
}


/***********************Sudoku Board Section    ***********************/

#container {
	height: auto;
	width: 540px;
	background-color: black;
	display: flex;
	flex-wrap: wrap;
	justify-content: space-evenly;
```

```
        align-content: space-evenly;
        margin: 0 auto;
        border-radius: 40px;
}

#generate-sudoku {
        margin: auto;
        display: block;
}

#solve {
        margin: auto;
        display: block;
}

#container div {
        background-color: skyblue;
        height: 60px;
        width: 60px;
        box-sizing: border-box;
        font-family: sans-serif;
        text-align: center;
        vertical-align: middle;
        line-height: 60px;
        font-size: 30px;
        color: green;
        border-radius: 20px;
        box-shadow: 2px 2px 2px 2px white;
}

#container div:hover {
        background-color: yellow;
}

.lsb {
        border-left: black 6px solid;
}

.bsb {
        border-bottom: black 6px solid;
}

.rsb {
        border-right: black 6px solid;
}

.tsb {
        border-top: black 6px solid;
}
```

```
.ldb {
	border-left: black 1px dashed;
}

.bdb {
	border-bottom: black 1px dashed;
}

.rdb {
	border-right: black 1px dashed;
}

.tdb {
	border-top: black 1px dashed;
}


/**************************Button Section ************************/

.btn {
	background-color: hotpink;
	border: 3px solid black;
	font-size: 20px;
	text-shadow: 2px 2px #ff0000;
	border-radius: 20px;
	font-weight: bolder;
	box-shadow: 2px 2px 2px 2px white;
}
```

JavaScript:

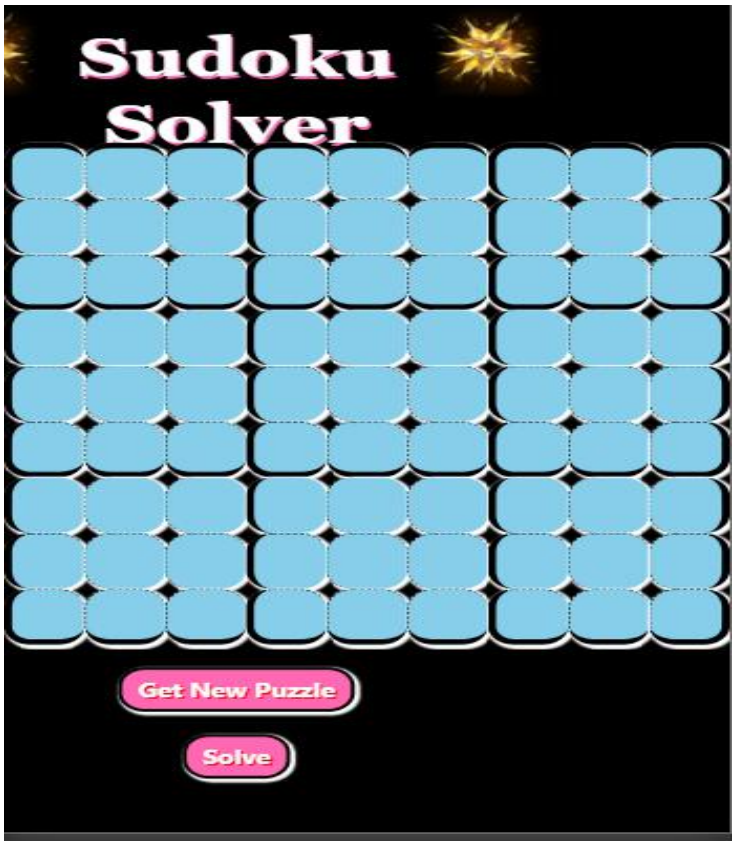
```
var arr = [
	[],
	[],
	[],
	[],
	[],
	[],
	[],
	[],
	[]
]
var temp = [
	[],
	[],
	[],
	[],
	[],
	[],
	[],
	[],
	[]
]

for (var i = 0; i < 9; i++) {
	for (var j = 0; j < 9; j++) {
		arr[i][j] = document.getElementById(i * 9 + j);

	}
}

function initializeTemp(temp) {

	for (var i = 0; i < 9; i++) {
		for (var j = 0; j < 9; j++) {
			temp[i][j] = false;

		}
	}
}


function setTemp(board, temp) {

	for (var i = 0; i < 9; i++) {
		for (var j = 0; j < 9; j++) {
			if (board[i][j] != 0) {
				temp[i][j] = true;
```

```
                }

            }
        }
}


function setColor(temp) {

        for (var i = 0; i < 9; i++) {
            for (var j = 0; j < 9; j++) {
                if (temp[i][j] == true) {
                    arr[i][j].style.color = "#DC3545";
                }

            }
        }
}

function resetColor() {

        for (var i = 0; i < 9; i++) {
            for (var j = 0; j < 9; j++) {

                arr[i][j].style.color = "green";


            }
        }
}

var board = [
        [],
        [],
        [],
        [],
        [],
        [],
        [],
        [],
        []
]


let button = document.getElementById('generate-sudoku')
let solve = document.getElementById('solve')

console.log(arr)

function changeBoard(board) {
```

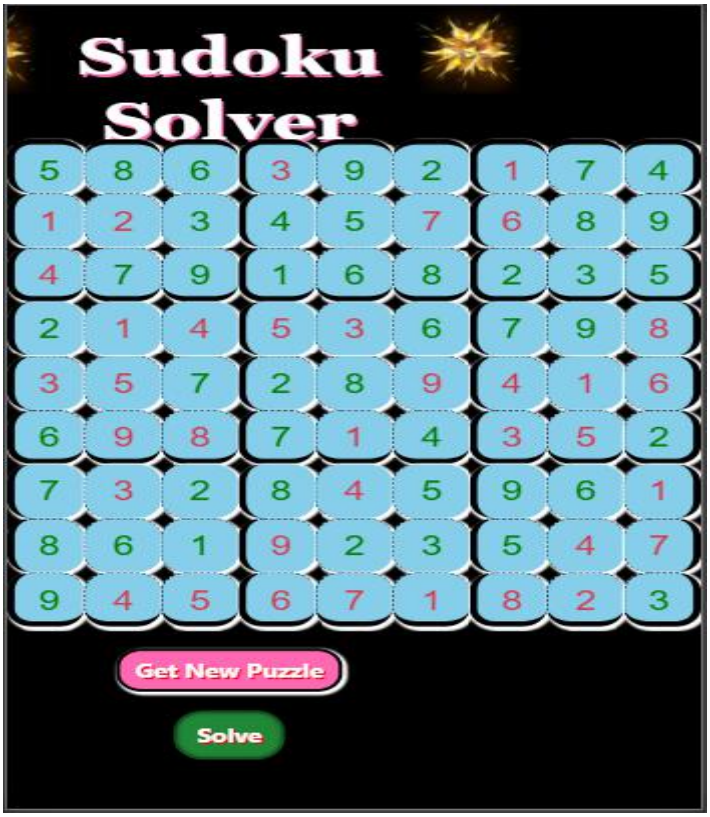
```
	for (var i = 0; i < 9; i++) {
		for (var j = 0; j < 9; j++) {
			if (board[i][j] != 0) {

				arr[i][j].innerText = board[i][j]
			} else
				arr[i][j].innerText = ''
		}
	}
}


button.onclick = function() {
	var xhrRequest = new XMLHttpRequest()
	xhrRequest.onload = function() {
		var response = JSON.parse(xhrRequest.response)
		console.log(response)
		initializeTemp(temp)
		resetColor()

		board = response.board
		setTemp(board, temp)
		setColor(temp)
		changeBoard(board)
	}
	xhrRequest.open('get', 'https://sugoku.herokuapp.com/board?difficulty=easy')
		//we can change the difficulty of the puzzle the allowed values of difficulty
are easy, medium, hard and random
	xhrRequest.send()
}

//to be completed by student, function should not return anything
// you can make a call to changeboard(board) function to update the state on the
screen
//returns a boolean true of false

function isSafe(board, r, c, no) {


	//not repeating in the same row or column
	for (var i = 0; i < 9; i++) {
		if (board[i][c] == no || board[r][i] == no) {
			return false;
		}
	}
	//subgrid
	var sx = r - r % 3;
	var sy = c - c % 3;

	for (var x = sx; x < sx + 3; x++) {
```

```
            for (var y = sy; y < sy + 3; y++) {
                if (board[x][y] == no) {
                    return false;
                }
            }
        }

        return true;
}

function solveSudokuHelper(board, r, c) {

        //base case
        if (r == 9) {
            changeBoard(board);
            return true;
        }
        //other cases
        if (c == 9) {
            return solveSudokuHelper(board, r + 1, 0);
        }
        //pre-filled cell, skip it
        if (board[r][c] != 0) {
            return solveSudokuHelper(board, r, c + 1);
        }

        //there is 0 in the current location
        for (var i = 1; i <= 9; i++) {

            if (isSafe(board, r, c, i)) {
                board[r][c] = i;
                var success = solveSudokuHelper(board, r, c + 1);
                if (success == true) {
                    return true;
                }
                //backtracking step
                board[r][c] = 0;
            }

        }
        return false;

}
function solveSudoku(board) {
        solveSudokuHelper(board, 0, 0);
}


solve.onclick = function() {
        solveSudoku(board)
```

}


Sudoku
Solver
Get New Puzzle
Solve


Sudoku
Solver
3 1
1 2 7 6
4
1 4 5 3 8
3 5 9 4 1 6
9 8 1 3 5
3 4 1
9 4 7
4 5 6 7 1 8 2
Get New Puzzle
Solve


Sudoku Solver
5 8 6 3 9 2 1 7 4
1 2 3 4 5 7 6 8 9
4 7 9 1 6 8 2 3 5
2 1 4 5 3 6 7 9 8
3 5 7 2 8 9 4 1 6
6 9 8 7 1 4 3 5 2
7 3 2 8 4 5 9 6 1
8 6 1 9 2 3 5 4 7
9 4 5 6 7 1 8 2 3
Get New Puzzle
Solve


Sudoku Solver
1 3 7 2 8 5 9 4 6
2 4 5 1 6 9 3 7 8
6 8 9 3 4 7 1 2 5
3 2 1 4 5 6 7 8 9
4 5 8 7 9 1 2 6 3
7 9 6 8 2 3 4 5 1
5 1 3 6 7 2 8 9 4
8 6 2 9 1 4 5 3 7
9 7 4 5 3 8 6 1 2
Get New Puzzle
Solve

Conclusion:

• A person can understand the logic behind the Sudoku Solver puzzle and try to solve Sudoku Puzzles anywhere he is needed to solve it.

• Back tracking algorithm takes less memory and time compared to Dancing links algorithm.

• A solution is guaranteed (as long as the puzzle is valid)

• Solving time is mostly unrelated to degree of difficulty

We can set the Level of Difficulty of the Sudoku Solver Problem.There are three levels of difficulty which includes Easy ,Medium, Hard. However in general, for the testing were extracted from the web, work efficiently when algorithm is applied on it. These testing may contain problems which got solved and which not got solved.

References

- Geeks for Geeks.
- Coding Blocks.
- FreeCodeCamp.org.