

**E & TC Department**

**A.Y. 2022-23 Semester-I**

**Design analysis and algorithms**

**Subject :-**

**Course Project – Sudoku Solver (Backtracking)**

**Presented By:-**

|  |  |  |
| --- | --- | --- |
| **Sr No** | **Student Name** | **Roll No** |
| **1.** | **Swapnil Patil** | **59** |
| **2.** | **Komal Shinde** | **69** |
| **3.** | **Devyani Ushir** | **75** |
| **4.** | **Aditya Vhanmane** | **76** |



**INDEX**

|  |  |  |
| --- | --- | --- |
| **SR.No** | **TOPIC** | **Page No.** |
| **1** | **Introduction** | **3** |
| **2** | **Backtracking Algorithm** | **4** |
| **3** | **Algorithm For Sudoku Solver** | **6** |
| **4** | **Sample input & Output** | **7-8** |
| **5** | **Advantages , Disadvantages** | **8-9** |
| **6** | **Application** | **9** |
| **7** | **Future Scope** | **10** |
| **8** | **Conclusion** | **10** |

**Problem Statement:**

People nowadays always busy for working and cope for the challenges on their life. With the increasing busyness of the life, stress become a common issue that faced by the society. A long term of facing a particular problem will make a people to feel pressure and stress themselves badly. Therefore, they need some break to take their mind off from working materials before they continue to work. Sometimes, a challenges mini game like Sudoku can helps to exercise their brain and reduce stress.

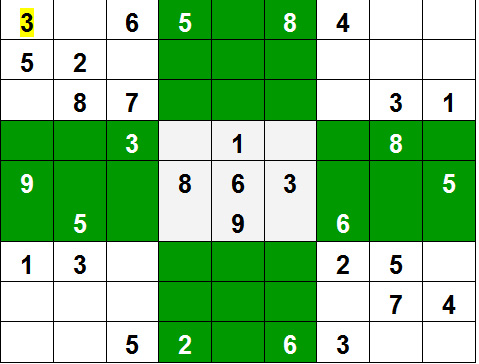
**OBJECTIVES**

Understanding the algorithm behind Sudoku puzzle. Identify an algorithm to generate a complete Sudoku puzzle with the rules that the digit 1-9 appear exactly once in each row, each columns, and each 3×3 sub grids. Identify the ways of removing cells from the complete Sudoku puzzle to create a valid game. Generate Sudoku with different difficulty level. Identify the algorithm to solve the Sudoku puzzle.

The purpose is to implement a more efficient algorithm and then compare it with another Sudoku solver named as brute force algorithm.

1. **INTRODUCTION**

Like all Other [Backtracking problems](https://www.geeksforgeeks.org/archives/tag/backtracking), Sudoku can be solved by assigning numbers one by one to empty cells. Before assigning a number, check whether it is safe to assign. Check that the same number is not present in the current row, current column and current 3X3 sub grid. After checking for safety, assign the number, and recursively check whether this assignment leads to a solution or not. If the assignment doesn’t lead to a solution, then try the next number for the current empty cell. And if none of the number (1 to 9) leads to a solution, return false and print no solution exists.



1. **What is Backtracking:**

Backtracking is a kind of brute-force approach which comes into picture when solving a problem requires considering multiple choices as we don't know which choice is correct and we try to solve the problem using trail and error method considering one choice at a time until required answer is obtained.The difference between brute-force and backtracking is that in backtracking, we do not explore all the possible options instead we explore only worthy options and build the solution incrementally.

**2.1 Backtracking Algorithm:**

Backtrack(s)

    ifs is not a solution

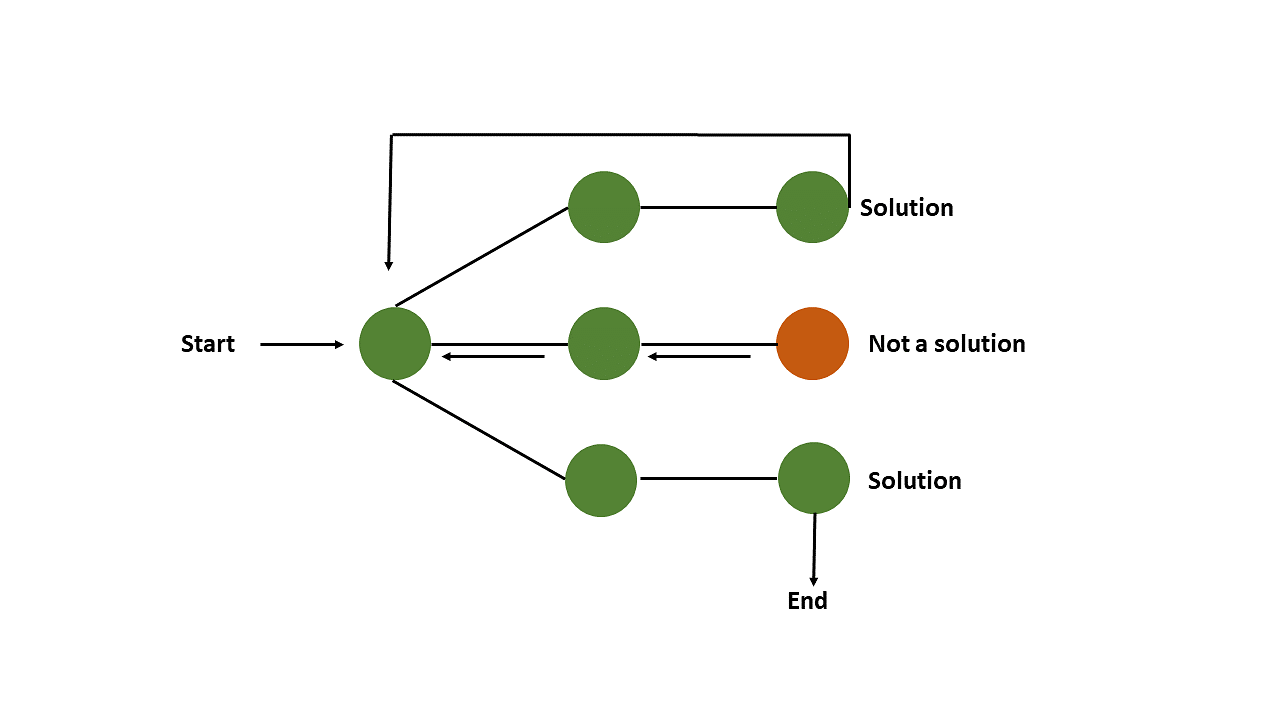
 return false

    if is a new solution

        add to list of solutions

    backtrack(expand s)

It finds a solution by building a solution step by step, increasing levels over time, using [recursive calling](https://www.simplilearn.com/tutorials/data-structure-tutorial/recursive-algorithm). A search tree known as the state-space tree is used to find these solutions. Each branch in a state-space tree represents a variable, and each level represents its solution.



1. **A sudoku solution must satisfy all of the following rules:**
2. Each of the digits 1-9 must occur exactly once in each row.
3. Each of the digits 1-9 must occur exactly once in each column.
4. Each of the digits 1-9 must occur exactly once in each of the 9 3x3 sub-boxes of the grid.
5. The '.' character indicates empty cells.
6. **Algorithm For Sudoku Solver:**
7. Find an unfilled cell (i,j) in grid.

2. If all the cells are filled then.

2.1. A valid sudoku is obtained hence return true.

3. For each num in 1 to 9.

3.1. If the cell (i,j) can be filled with num then fill it with num temporarily to

check.

3.2. If sudoku Solver(grid) is true then return true.

3.3. If the cell (i, j) can't be filled with num the mark it as unfilled to trigger

backtracking.

4. If none of the numbers from 1 to 9 can be filled in cell (i,j) then return false as

there is no solution for this sudoku.

1. A sudoku solution must satisfy all of the following rules:
   1. Each of the digits 1-9 must occur exactly once in each row.
   2. Each of the digits 1-9 must occur exactly once in each column.
   3. Each of the digits 1-9 must occur exactly once in each of the 9 3x3 sub-boxes of the grid.
   4. The '.' character indicates empty cells.

* **Time complexity:**

Consider there are m unfilled cells in the sudoku

Time Complexity:

The backtracking algorithm takes **O(9m)** time complexity in the worst case since for every unfilled cell there are **9** possibilities to explore and there are **m** unfilled cells in the sudoku.

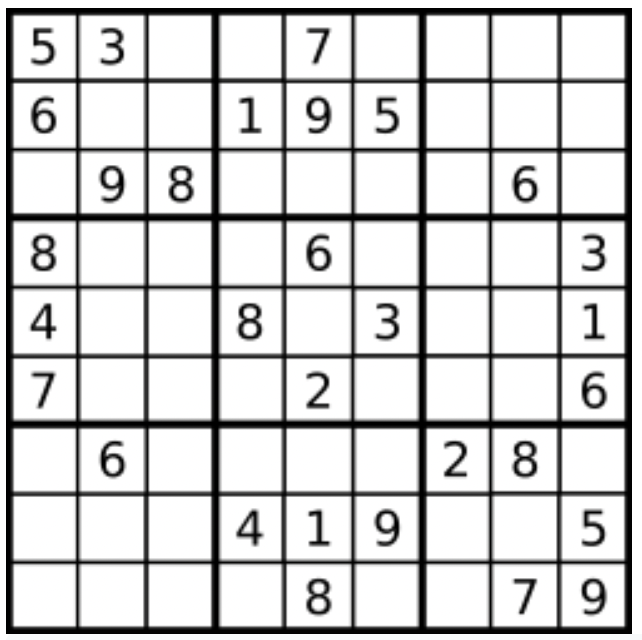
* Worst Case Time Complexity: O(9m)
* Average Case Time Complexity: O(9m)
* Best Case Time Complexity: O(m2) [This takes place when the number of backtracking steps are minimized]

O(9(N\*N)), For every unassigned index, there are 9 possible options so the time complexity is O(9^(n\*n)). The time complexity remains the same but there will be some early pruning so the time taken will be much less than the naive algorithm but the upper bound time complexity remains the same.

* **Space Complexity:** O(N\*N), To store the output array a matrix is needed.

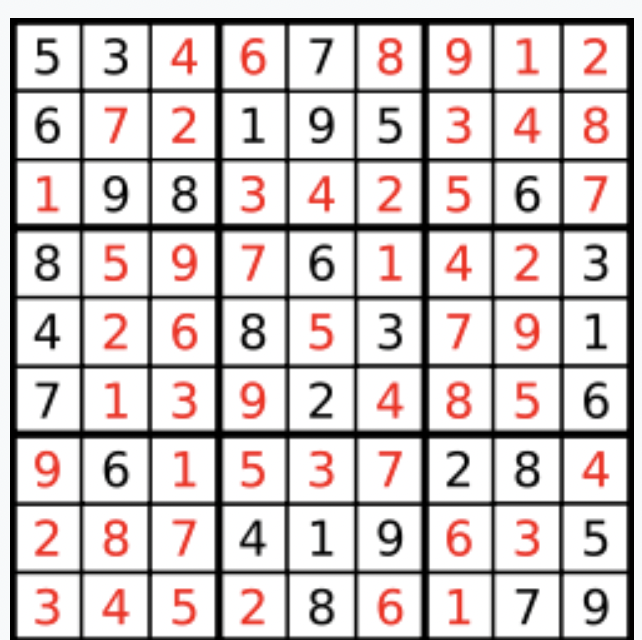
At most, there can be **m** function calls in the recursion stack. Hence the space complexity would be **O(m)**

1. **Sample input:**

****

Board= [["5","3",".",".","7",".",".",".","."],["6",".",".","1","9","5",".",".","."],[".","9","8",".",".",".",".","6","."],["8",".",".",".","6",".",".",".","3"],["4",".",".","8",".","3",".",".","1"],["7",".",".",".","2",".",".",".","6"],[".","6",".",".",".",".","2","8","."],[".",".",".","4","1","9",".",".","5"],[".",".",".",".","8",".",".","7","9"]]

1. **Sample Output:**

****

[["5","3","4","6","7","8","9","1","2"],["6","7","2","1","9","5","3","4","8"],["1","9","8","3","4","2","5","6","7"],["8","5","9","7","6","1","4","2","3"],["4","2","6","8","5","3","7","9","1"],["7","1","3","9","2","4","8","5","6"],["9","6","1","5","3","7","2","8","4"],["2","8","7","4","1","9","6","3","5"],["3","4","5","2","8","6","1","7","9"]]

1. **Advantages of solving a sudoku:**
2. Sudoku boosts logical thinking.
3. Playing sudoku improves Memory and recall.
4. Sudoku helps to developed quick thinking skills.
5. Playing sudoku helps to improve concentration.
6. It helps to reduce anxiety and stress.

1. **Disadvantages of playing sudoku:**
2. Addiction.
3. Time consuming.
4. Sudoku is threat to Crossword.
5. Sense of incompletion.
6. Frustration.
7. **Real time applications of sudoku solving:**
8. Sudoku Championship
9. Helps with clarity of mind
10. Artificial Intelligence
11. Mathematics
12. Spy-ware
13. **Future Scope:**
14. Scope of the project is generating the Sudoku puzzle with different difficulty level and solves the Sudoku game. The scope covers:
15. Allow user to select the difficulty level of the Sudoku game.
16. Generate a complete Sudoku puzzle by following the Sudoku rules which are the digits 1-9 appear exactly once in each row, each columns, and each 3×3 sub grids.
17. Remove cells to create a valid Sudoku games according to the difficulty level that the user selected.
18. Allow user to input the digit 1-9 into the empty cells of the puzzle.
19. Highlight the incorrect input of digit with red colour.
20. Solve the Sudoku games.

**Conclusion:**

Backtracking algorithm is an appropriate method to find a solution faster and more efficient compared to the brute force algorithm. The proposed algorithm is able to solve such puzzles with any level of difficulties in a short period of time (less than one second). A Sudoku (top) being solved by backtracking. Each cell is tested for a valid number, moving "back" when there is a violation, and moving forward again until the puzzle is solved. A Sudoku designed to work against the brute force algorithm.