Sudoku Solver LeetCode

Write a program to solve a Sudoku puzzle by filling the empty cells.

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.

The '.' character indicates empty cells.

Example: The Input Sudoku:

5 3 . . 7 . . . .

6 . . 1 9 5 . . .

. 9 8 . . . . 6 .

8 . . . 6 . . . 3

4 . . 8 . 3 . . 1

7 6

. 6 . . . . 2 8 .

. . . 4 1 9 . . 5

. . . . 8 . . 7 9

The Solved Sudoku Board:

5 3 4 6 7 8 9 1 2

6 7 2 1 9 5 3 4 8

1 9 8 3 2 4 5 6 7

8 1 9 7 6 2 4 5 3

4 2 6 8 5 3 7 9 1

7 5 3 9 4 1 8 2 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

Approach 1: Function to solve the Sudoku puzzle

Function Purpose:

Solve a Sudoku puzzle using the backtracking approach.

Explanation:

- **isPossible Function:**
    - Checks if placing a value at a specific position is valid in the Sudoku grid.
    - Validates the value in the current row, column, and the 3x3 subgrid.
- **solve Function:**
    - Recursive backtracking function to explore all possible placements of values on the Sudoku grid.
    - Tries placing values from '1' to '9' at empty cells.
    - Checks for the validity of the placement using the **isPossible** function.
    - If a solution is found, returns true; otherwise, backtracks.
- **solveSudoku Function:**
    - Initializes the Sudoku solving process by calling the **solve** function.

Time Complexity:

- **The time complexity is O(9^M), where M is the number of empty cells in the Sudoku grid.**

Space Complexity:

- **Sudoku Grid Storage: O(N^2), where N is the size of the Sudoku grid.**