[Click on the logo to find **Problem Statement**]

Intuition

This code checks the validity of a Sudoku board using three defaultdicts (`row` , `column` , and `box`) to track unique number occurrences in rows, columns, and 3x3 boxes. It iterates through the 9x9 board, skipping empty cells, and returns False if a number violates Sudoku rules. Sets in defaultdicts efficiently prevent duplicates. If the entire board is traversed without conflicts, it returns True, confirming a valid Sudoku configuration where each row, column, and box contains only unique numbers.

Approach

1. **Data Structures:**
    - Utilizes three defaultdicts (`row` , `column` , and `box`) with sets to keep track of unique occurrences of numbers in each row, column, and 3x3 box of the Sudoku board.
2. **Iteration:**
    - Uses nested loops to iterate over each cell of the 9x9 Sudoku board.
3. **Skip Empty Cells:**
    - Skips over cells with "." (indicating empty) as they don't contribute to rule violations.
4. **Check Violations:**
    - Checks if the current number violates Sudoku rules by looking for its presence in the corresponding row, column, and box sets.
5. **Return False on Violation:**
    - If a violation is detected, returns False, indicating that the board is not valid.
6. **Update Sets:**
    - If the number is not violating any rules, updates the sets in `row` , `column` , and `box` defaultdicts to record its presence.

Complexity

- Time complexity: $O(n^2)$
- Space complexity: O(n)

Code

```
class Solution:
    def isValidSudoku(self, board: List[List[str]]) -> bool:
        row = collections.defaultdict(set)
        column = collections.defaultdict(set)
        box = collections.defaultdict(set)

        for i in range(9):
            for j in range(9):
                if board[i][j] == ".":
                    continue
                if (board[i][j] in row[i] or
                    board[i][j] in column[j] or
                    board[i][j] in box[(i // 3 , j // 3)]):
                    return False
                row[i].add(board[i][j])
                column[j].add(board[i][j])
                box[(i // 3 , j // 3)].add(board[i][j])
        return True
```

If you want to see more solutions to coding problems, you can visit: