LeetCode 36 [Valid Sudoku]

Determine if a 9 x 9 Sudoku board is valid.

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

We need to take care of Row, Cols and Boxes.

Row

1	.	3	.	6	.	2	.	.

valid

HashSet

1
3
6
2

we return false if is duplicated

We create 9 HS for rows

Colums
Is the same logis from rows

Boxes

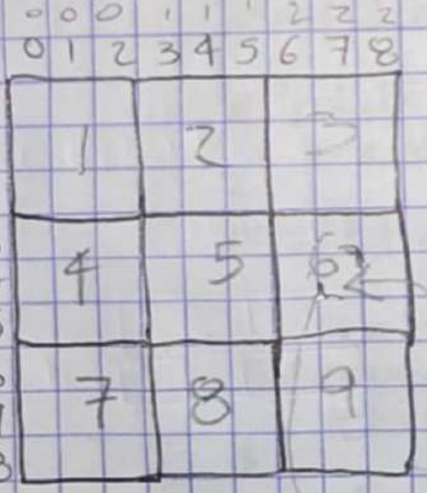
9 hashes

Float

$0/3 = 0$

$1/3 = 0$

$2/3 = 0$



Formula to determine box number

$$\boxed{\frac{R}{3} + 3 + \frac{C}{3}}$$

$\frac{4}{3} + 3 + \frac{7}{3}$

$= 1 + 3 + 2$

$= 6 \longrightarrow$ We check at HS 6.

Time: $O(n^2)$?!

Space: $O(3n) = O(n)$??

or

$O(1)$

Java Solution

```
public boolean isValidSudoku(char[][] board) {
    HashSet<Character>[] rows = new HashSet[9];
    HashSet<Character>[] cols = new HashSet[9];
    HashSet<Character>[] boxes = new HashSet[9];

    For(int r=0; r<9; r++) {
        rows[r] = new HashSet<Character>();
        cols[r] = new HashSet<Character>();
        boxes[r] = new HashSet<Character>();
    }

    For(int r=0; r<9; r++){
        For(int c =0; c<9; c++) {
            char val = board[r][c];

            if(val == '.') {
                continue;
            }

            if(rows[r].contains(val)) {
                return False;
            }
            rows[r].add(val);

            if(cols[c].contains(val)) {
                return False;
            }
            cols[c].add(val);

            int boxNumber = (r/3)*3 + c/3;
            if(boxes[boxNumber].contains(val)) {
                return False;
            }
            boxes[boxNumber].add(val);
        }
    }

    return true;

}
```

Kotlin solution

```
Fun isValidSudoku (board: Array<CharArray>): Boolean {
    val rows = Array(9) { HashSet<Char>() }
    val cols = Array(9) { HashSet<Char>() }
    val boxes = Array(9) { HashSet<Char>() }

    For(r in 0 until 9) {
        For( c in 0 until 9) {
            val value = board[r][c]

            if(value == '.') continue

            if(rows[r].contains(value)) {
                return False
            }
            rows[r].add(value)

            if(cols[c].contains(value)) {
                return False
            }
            cols[c].add(value)

            val boxIndex = (r/3)*3 + (c/3)
            if(boxes[boxIndex].contains(value)) {
                return false
            }
            boxes[boxIndex].add(value)
        }
    }

    return true
}
```