Valid sudoku - hash set (one pass)

```
class Solution:
    def is_valid_sudoku(self, board: List[List[str]]) -> bool:

        cols = collections.defaultdict(set)
        rows = collections.defaultdict(set)
        squares = collections.defaultdict(set)

        for r in range(9):
            for c in range(9):
                if board[r][c] == ".":
                    continue
                if (board[r][c] in rows[r] or
                    board[r][c] in cols[c] or
                    board[r][c] in squares[(r//3, c//3)]):
                    return False

                cols[c].add(board[r][c])
                rows[r].add(board[r][c])
                squares[(r//3, c//3)].add(board[r][c])
```

return True

```
print(Solution().is_valid_sudoku(board =
[["1","2",".",".","3",".",".",".","."],
 ["4",".",".","5",".",".",".",".","."],
 [".","9","8",".",".",".",".",".","3"],
 ["5",".",".",".","6",".",".",".","4"],
 [".",".",".","8",".","3",".",".","5"],
 ["7",".",".",".",".","2",".",".","6"],
 [".",".",".",".",".",".","2",".","."],
 [".",".",".","4","1","9",".",".","8"],
 [".",".",".",".","8",".",".","7","9"]]))
```

This exercise is using hash set (one pass) algorithm, and is running in approximately 357 steps.

Step 1:

cols, rows, squares -> defaultdict(<class 'set'>, {})

Step 2: based on the position of our values in the sudoku, when the division happens in our variable squares, the result is always rounded.

BIG Step 3:

```
for r in range(9):
    for c in range(9):
        if board[r][c] == '.':
            continue
        if (board[r][c] in rows[r] or
            board[r][c] in cols[c] or
            board[r][c] in squares[(r//3, c//3)]
            return False
```

- when r -> 0 then the iteration happens for the next row where c becomes from 0 to 8 and then r goes 1. and so on and so forth.

BIG Step 4:

```
cols[c].add(board[r][c])
rows[r].add(board[r][c])
squares[(r//3, c//3)].add(board[r][c])
```

when r -> 0 and c -> 0, 1, 2, ... 7, 8:

cols -> defaultdict(<class 'set'>, {0: {'1', '6'}, 1: {'2', '4'}, 4: {'3', '7'}})

rows -> defaultdict(<class 'set'>, {0: {'3', '1', '2', '6'}})

squares -> defaultdict(<class 'set'>, {(0,0): {'1', '2', '6'}, (0,1): {'3', '4'}})

and so on and so forth, until r -> 8 and c -> 8

cols -> {0: {'5', '4', '7', '1', '6'}, 1: {'9', '2'}, 4: {'1', '2', '3', '6', '8'},

3: {5, 4, 8}, 2: {8}, 8: {3, 5, 9, 4, 6, 8}, 5: {3, 9},
6: {2}, 7: {7}})

rows → {0: {3, 1, 2}, 1: {5, 4}, 2: {9, 3, 8}, 3: {5, 4, 6},
4: {3, 5, 8}, 5: {6, 7, 2}, 6: {2}, 7: {9, 4, 8, 1},
8: {8, 7, 8}})

squares → {(0,0): {1, 2, 9, 4, 8}, (0,1): {3, 5}, (0,2): {3}, (1,0):
{5, 7}, (1,1): {3, 2, 6, 8}, (1,2): {5, 4, 6}, (2,2):
{9, 7, 2, 8}, (2,1): {9, 4, 8, 1}})

this validates that there aren't any duplicates. return True ✓