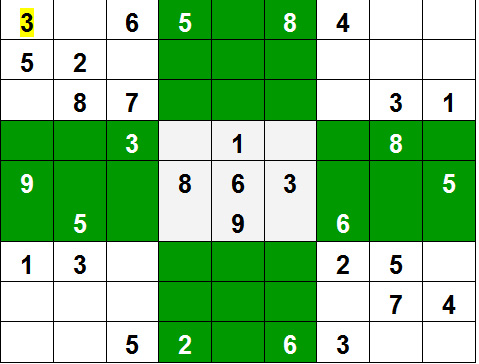
Given a partially filled 9×9 2D array ‘grid[9][9]’, the goal is to assign digits (from 1 to 9) to the empty cells so that every row, column, and subgrid of size 3×3 contains exactly one instance of the digits from 1 to 9.



**Example:**

|  |
| --- |
| **Input:**  grid = { {3, 0, 6, 5, 0, 8, 4, 0, 0},  {5, 2, 0, 0, 0, 0, 0, 0, 0},  {0, 8, 7, 0, 0, 0, 0, 3, 1},  {0, 0, 3, 0, 1, 0, 0, 8, 0},  {9, 0, 0, 8, 6, 3, 0, 0, 5},  {0, 5, 0, 0, 9, 0, 6, 0, 0},  {1, 3, 0, 0, 0, 0, 2, 5, 0},  {0, 0, 0, 0, 0, 0, 0, 7, 4},  {0, 0, 5, 2, 0, 6, 3, 0, 0} }  **Output:**  3 1 6 5 7 8 4 9 2  5 2 9 1 3 4 7 6 8  4 8 7 6 2 9 5 3 1  2 6 3 4 1 5 9 8 7  9 7 4 8 6 3 1 2 5  8 5 1 7 9 2 6 4 3  1 3 8 9 4 7 2 5 6  6 9 2 3 5 1 8 7 4  7 4 5 2 8 6 3 1 9 |

**Explanation:** Each row, column and 3\*3 box of

the output matrix contains unique numbers.

**Method 1:** Simple.  
**Approach:** The naive approach is to generate all possible configurations of numbers from 1 to 9 to fill the empty cells. Try every configuration one by one until the correct configuration is found, i.e. for every unassigned position fill the position with a number from 1 to 9. After filling all the unassigned position check if the matrix is safe or not. If safe print else recurs for other cases.

**Algorithm:**

1. Create a function that checks if the given matrix is valid sudoku or not. Keep Hashmap for the row, column and boxes. If any number has a frequency greater than 1 in the hashMap return false else return true;
2. Create a recursive function that takes a grid and the current row and column index.
3. Check some base cases. If the index is at the end of the matrix, i.e. i=N-1 and j=N then check if the grid is safe or not, if safe print the grid and return true else return false. The other base case is when the value of column is N, i.e j = N, then move to next row, i.e. i++ and j = 0.
4. if the current index is not assigned then fill the element from 1 to 9 and recur for all 9 cases with the index of next element, i.e. i, j+1. if the recursive call returns true then break the loop and return true.
5. if the current index is assigned then call the recursive function with index of next element, i.e. i, j+1

**Complexity Analysis:**

* **Time complexity:** O(9^(n\*n)).   
  For every unassigned index, there are 9 possible options so the time complexity is O(9^(n\*n)).
* **Space Complexity:** O(n\*n).   
  To store the output array a matrix is needed.