BIRZEIT UNIVERSITY

Design and Analysis of Algorithms (COM336)

Fall Semester 2020/2021

Project # 4

Due Date (Project#3 + Project #4): 11,12-Jan-2021

Backtracking Algorithm

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 sub grid of size 3×3 contains exactly one instance of the digits from 1 to 9.

3		6	5		8	4		
5	2							
	8	7					3	1
		3		1			8	
9	5		8	6 9	3	6		5
1	3					2	5	
							7	4
		5	2		6	3		

Method: Backtracking.

Approach:

Like all other Backtracking problems, Sudoku can be solved by one by one assigning numbers to empty cells. Before assigning a number, check whether it is safe to assign. Check that the same number is not present in the current row, current column and current 3X3 sub grid. After checking for safety, assign the number, and recursively check whether this assignment leads to a solution or not. If the assignment doesn't lead to a solution, then try the next number for the current empty cell. And if none of the number (1 to 9) leads to a solution, return false and print no solution exists.

Algorithm:

1. Create a function that checks after assigning the current index the grid becomes unsafe or not. Keep HashMap for a row, column and boxes. If any number has a frequency greater than 1 in the HashMap return false else return true; HashMap can be avoided by using loops.
2. Create a recursive function that takes a grid.
3. Check for any unassigned location. If present then assign a number from 1 to 9, check if assigning the number to current index makes the grid unsafe or not, if safe then recursively call the function for all safe cases from 0 to 9. if any recursive call returns true, end the loop and return true. If no recursive call returns true then return false.
4. If there is no unassigned location then return true.

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.

Input:

grid = { { 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, 0, 1, 5, 9, 8, 7 },
{ 9, 7, 4, 8, 6, 0, 1, 2, 5 },
{ 8, 5, 1, 7, 9, 2, 6, 4, 3 },
{ 1, 3, 8, 0, 4, 7, 2, 0, 6 },
{ 6, 9, 2, 3, 5, 1, 8, 7, 4 },
{ 7, 4, 5, 0, 8, 6, 3, 1, 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.