Rat in a Maze (all paths)

Problem Level: Hard

Problem Description:

You are given a N*N maze with a rat placed at maze[0][0]. Find and print all paths that rat can follow to reach its destination i.e. maze[N-1][N-1]. Rat can move in any direction (left, right, up and down).
Value of every cell in the maze can either be 0 or 1. Cells with value 0 are blocked means rat cannot enter into those cells and those with value 1 are open.

Sample Input 1:

```
3
1 0 1
1 1 1
1 1 1
```

Sample Output 1:

```
1 0 0 1 1 1 1 1 1
1 0 0 1 0 0 1 1 1
1 0 0 1 1 0 0 1 1
1 0 0 1 1 1 0 0 1
```

Explanation:

```
4 paths are possible which are printed in the required format.
```

Approach to be followed:

For this code, the approach is quite intuitive. We will form a recursive function, which will follow a path. We will keep checking if this path successfully reaches the destinations while avoiding all the blocked cells. If this happens, we will print that particular path. If a particular path does not reach the destination then backtrack and try other paths.

Steps:

1. Create a matrix, **solution**, and initially fill it with 0s.

2. Create a recursive function, which takes the input matrix, output matrix and position of rat **(i, j)**.
3. Check if the position is outside the matrix, or at a blocked cell. If this happens, simply return from the function.
4. Also, check if this position has already been backtracked. If yes, return from the function.
5. Otherwise, mark the position **solution[i][j]** as one and check if the current position is the destination or not. If the destination is reached, print the output matrix and return.
6. Recursively call for position (i + 1, j) , (i, j + 1) , (i – 1, j) , (i, j – 1).
7. Unmark position (i, j), that is, **solution[i][j]** as zero**.**

Pseudo Code:

```
function ratInAMaze(maze , n):
    loop from i = 0 to i = n:
        loop from j = 0 to j = n:
            solution[i][j] = 0
    solveMaze(maze, solution, 0, 0 , n) // starting position = (0, 0)


function solveMaze(maze, solution, x, y , n):
    if x equals n - 1 and y equals n - 1:
        solution[x][y] = 1
        printSolution(solution , n)
        return
    if(x > n-1 or x < 0 or y > n - 1 or y < 0 or  maze[x][y] == 0 or
solution[x][y] == 1):
        return


    solution[x][y] = 1
    solveMaze(maze, solution, x - 1, y , n)
    solveMaze(maze, solution, x + 1, y , n)
    solveMaze(maze, solution, x , y - 1 , n)
    solveMaze(maze, solution, x, y + 1 , n)
    solution[x][y] = 0


function printSolution(solution , n):
    loop from i = 0 to i = n:
        loop from j = 0 to j = n:
            print solution[i][j]
```

Time Complexity: O(2^k), where **k** is the number of available nodes, which can be in the order of **N^2,** where N is the number of rows and columns in the matrix.