Code Link: LP_CODE3.java

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
```

Time Complexity: O(9^(n*n))

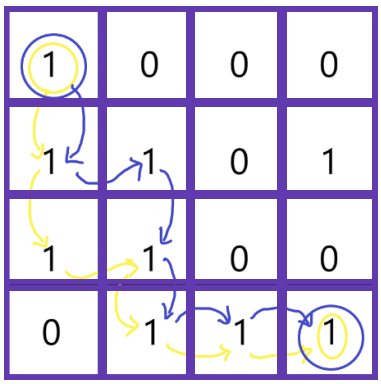
Space complexity : O(1)

Interview Problem: Rat in a Maze

Problem Statement: Consider the rat at (0, 0) in a square matrix of order N * N. The rat must reach its destination at (N - 1, N - 1). Find all possible paths for the rat to get from its source to its destination. The direction the mouse can move is "U" (up), "D" (down), "L" (left), and "R" (right). A matrix cell with a value of 0 means it is blocked and rats cannot pass through it, and a matrix cell with a value of 1 means that rats can pass through it.
Note: In a path, no cell can be visited more than one time.
Print the answer in lexicographical(sorted) order

Example 1:

Input:
N = 4
m[][] = {{1, 0, 0, 0},
{1, 1, 0, 1},
{1, 1, 0, 0},
{0, 1, 1, 1}}
Output: DDRDRR DRDDRR



The rat can reach the destination at (3, 3) from (0, 0) by two paths - DRDDRR and DDRDRR, when printed in sorted order we get DDRDRR DRDDRR.

Approach to Solve the problem
So we have a matrix. Next, find the path from the source cell (0,0) to the target cell(N-1, N-1) and do the following steps:

- Check the current cell. If this is the target cell, the puzzle is solved.
- If not, move down to see if it is possible to move in the downward direction (to move from a cell, the cell must not be free and in progress).
- We will continue the selected path down the next cell if we can move there.
- If not, let's move to the right cell if this is vacant and not already present in path.
- If not Move up when blocked or removed,
- Move to the cell on the left if it cannot move up.

If none of the four moves (down, right, up, or left) are possible, simply go back and change the current path (return).
Thus, the summary is that we try to move to the other cell (down, right, up, and left) from the current cell and if no movement is possible, then just come back and change the direction of the path to another cell.

Code: LP_CODE5.java

Output:

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

This is one of the valid paths for the above problem.

Next Class Teasers:

- Introduction to LinkedList
- Insertion and Deletion
- Reverse a linked list
- Cycle detection in a Linked List