1. **N Queens Problem**

class Solution {

public List<List<String>> solveNQueens(int n) {

List<List<String>> result = new ArrayList<>();

char[][] board = new char[n][n];

for (char[] row : board)

Arrays.fill(row, '.');

backtrack(0, board, result, n);

return result;

}

void backtrack(int row, char[][] board, List<List<String>> result, int n) {

if (row == n) {

List<String> current = new ArrayList<>();

for (char[] r : board)

current.add(new String(r));

result.add(current);

return;

}

for (int col = 0; col < n; col++) {

if (isSafe(board, row, col, n)) {

board[row][col] = 'Q';

backtrack(row + 1, board, result, n);

board[row][col] = '.';

}

}

}

boolean isSafe(char[][] board, int row, int col, int n) {

for (int i = 0; i < row; i++)

if (board[i][col] == 'Q') return false;

for (int i = row - 1, j = col - 1; i >= 0 && j >= 0; i--, j--)

if (board[i][j] == 'Q') return false;

for (int i = row - 1, j = col + 1; i >= 0 && j < n; i--, j++)

if (board[i][j] == 'Q') return false;

return true;

}

}

1. **Rat in a Maze**

class Solution {

public List<String> findPath(int[][] maze, int n) {

List<String> result=new ArrayList<>();

boolean[][] visited=new boolean[n][n];

if(maze[0][0]==1)

solve(0,0,maze,n,"",result,visited);

return result;

}

void solve(int x,int y,int[][] maze,int n,String path,List<String> result,boolean[][] visited) {

if(x==n-1 && y==n-1) {

result.add(path);

return;

}

visited[x][y]=true;

int[] dx={1,0,0,-1};

int[] dy={0,-1,1,0};

char[] move={'D','L','R','U'};

for (int i=0;i<4;i++) {

int newX=x+dx[i],newY=y+dy[i];

if(isSafe(newX,newY,maze,visited,n)) {

solve(newX,newY,maze,n,path+move[i],result,visited);

}

}

visited[x][y]=false;

}

boolean isSafe(int x,int y,int[][] maze,boolean[][] visited,int n) {

return x>=0 && y>=0 && x<n && y<n && maze[x][y]==1 && !visited[x][y];

}

}

1. Letter Combinations of a Phone Number

class Solution {

public List<String> letterCombinations(String digits) {

if (digits.isEmpty()){

return new ArrayList<>();

}

String[] mapping={

"", "", "abc", "def", "ghi",

"jkl", "mno", "pqrs", "tuv", "wxyz"

};

List<String> result=new ArrayList<>();

backtrack(0,digits,"",mapping,result);

return result;

}

void backtrack(int index,String digits,String path,String[] mapping,List<String> result) {

if(index==digits.length()) {

result.add(path);

return;

}

String letters=mapping[digits.charAt(index)-'0'];

for(char c : letters.toCharArray()) {

backtrack(index+1,digits,path+c,mapping,result);

}

}

}