Recursion - IV (Backtracking)

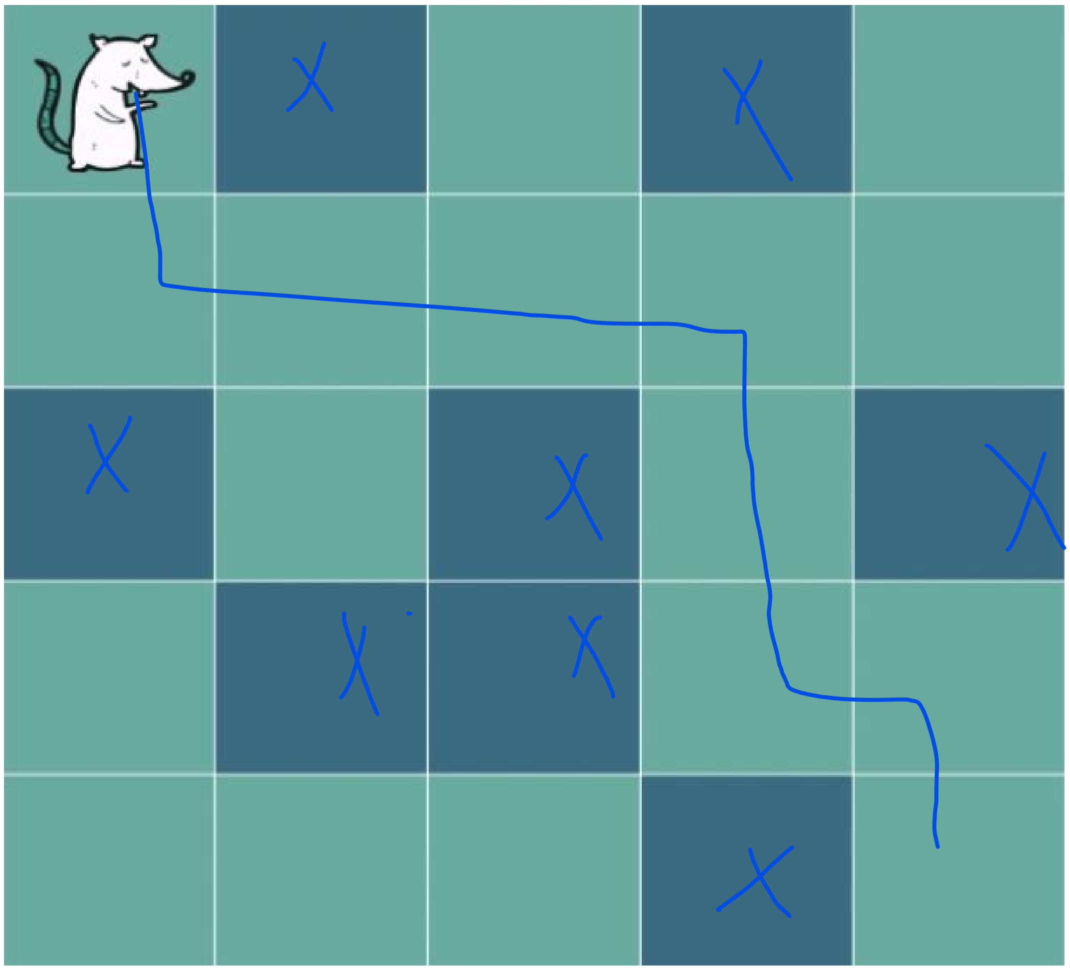
ma

Rat In a Maze

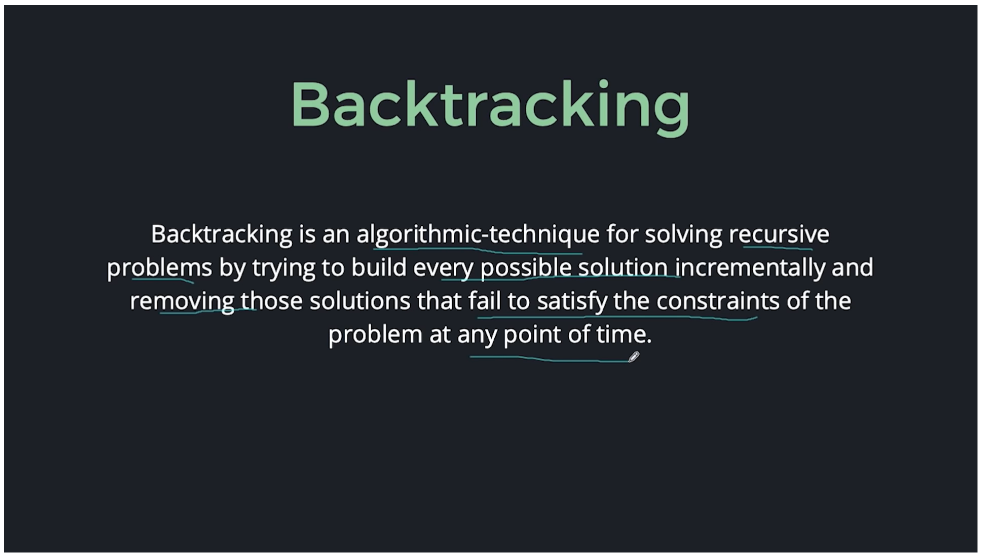
Problem: Given a maze(2D matrix) with obstacles, starting from (0,0) you have to reach (n-1, n-1). If you are currently on (x,y), you can move to (x+1,y) or (x,y+1). You can not move to the walls.

0,0 2X2

Idea: Try all the possible paths to see if you can reach (n-1,n-1)



n-1, n-1



Backtracking

Backtracking is an algorithmic-technique for solving recursive problems by trying to build every possible solution incrementally and removing those solutions that fail to satisfy the constraints of the problem at any point of time.

Input:

0 denotes wall, 1 denotes free path

two numbers n, m

Next n lines contain m numbers (0 or 1)

Output:

Print 1 if rat can reach (n-1,m-1)

Print 0 if it can not reach (n-1,m-1)

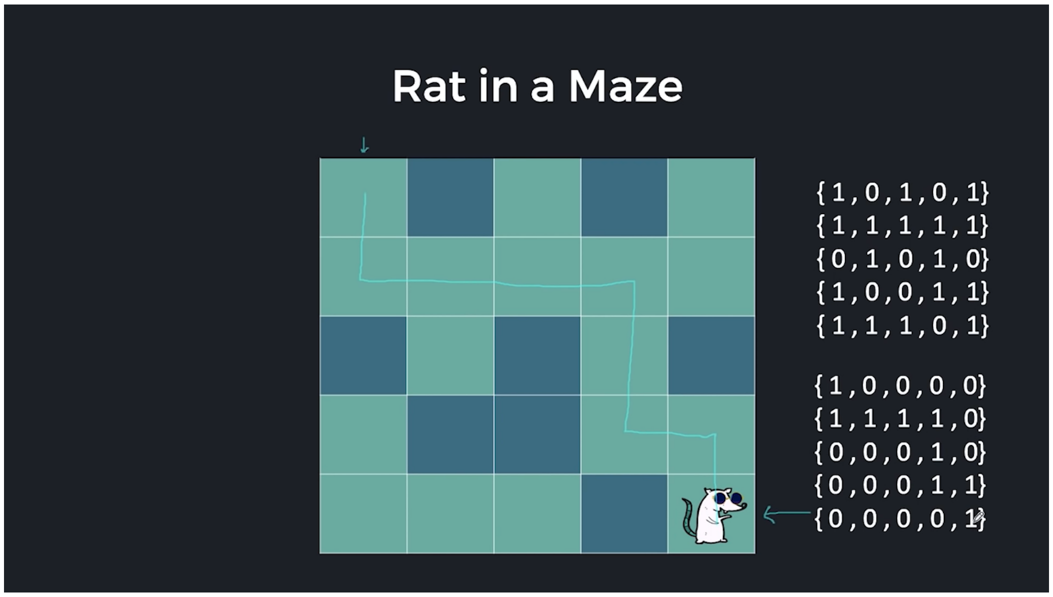
Test Case 1:

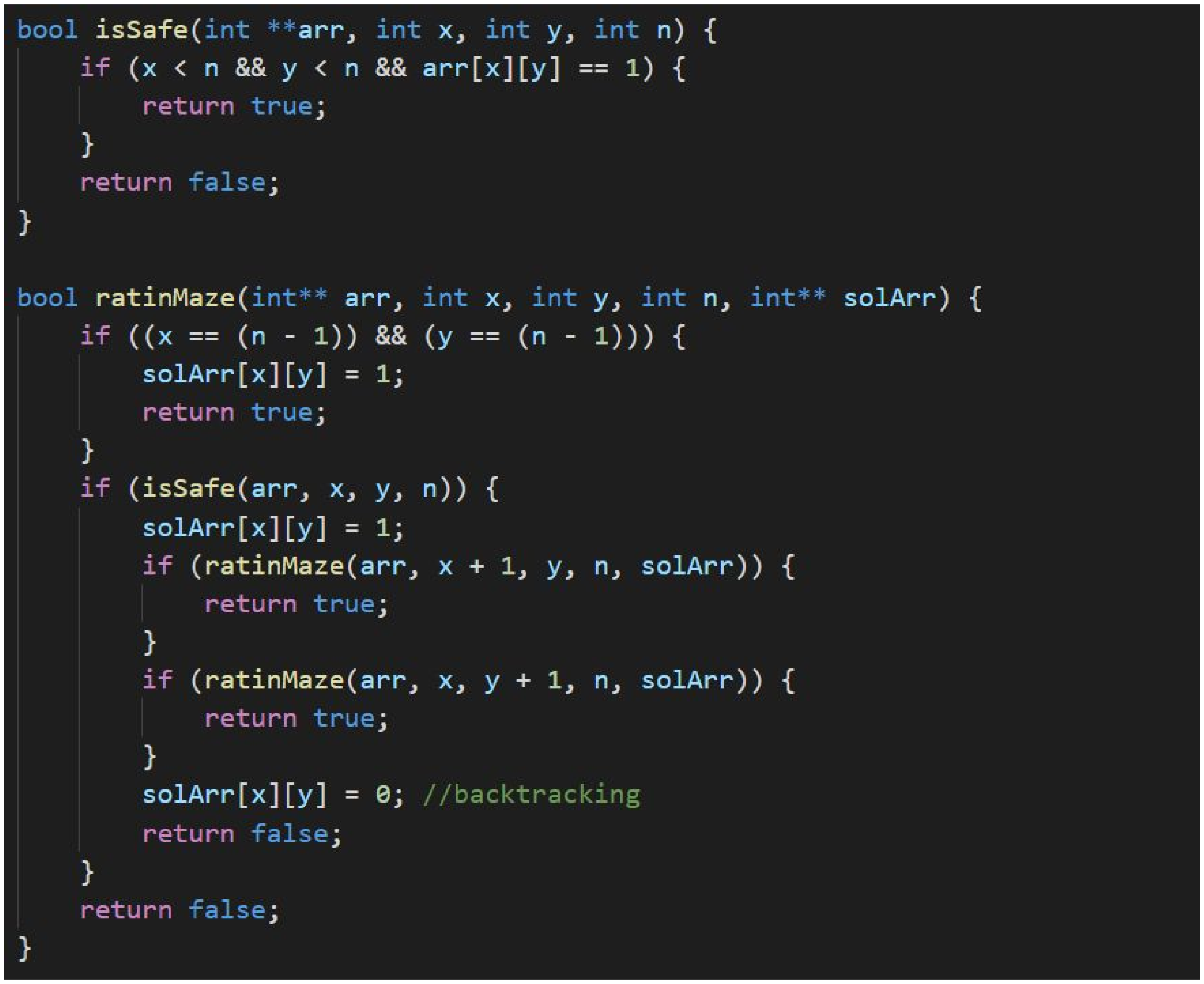
Input:

```
5 5
1 0 1 0 1
1 1 1 1 1
0 1 0 1 0
1 0 0 1 1
1 1 1 0 1
```

Output:

```
1
```




```
bool isSafe(int **arr, int x, int y, int n) {
    if (x < n && y < n && arr[x][y] == 1) {
        return true;
    }
    return false;
}

bool ratinMaze(int** arr, int x, int y, int n, int** solArr) {
    if ((x == (n - 1)) && (y == (n - 1))) {
        solArr[x][y] = 1;
        return true;
    }
    if (isSafe(arr, x, y, n)) {
        solArr[x][y] = 1;
        if (ratinMaze(arr, x + 1, y, n, solArr)) {
            return true;
        }
        if (ratinMaze(arr, x, y + 1, n, solArr)) {
            return true;
        }
        solArr[x][y] = 0; //backtracking
        return false;
    }
    return false;
}
```

Time Complexity: O(2^n)

Space Complexity: O(2^n)