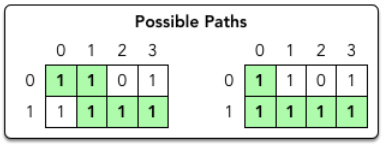
Problem Statement (/view_test_problem/218/131/) | Previous Submissions (/view_all_submissions_for_test_problem/218/131/)

Editorial | Top Submissions (/view_top_submissions/218/131/30/)

Problem Description

Consider a maze mapped to a matrix with an upper left corner at coordinates *(row, column) = (0, 0)*. Any movement must be in increasing row or column direction. You must determine the number of distinct paths through the maze. You will always start at position *(0, 0)*, the top left, and end up at *(max(row), max(column))*, the bottom right.
As an example, consider the following diagram where '`1`' indicates an open cell and '`0`' indicates blocked. You can only travel through open cells, so no path can go through the cell at *(0, 2)*. There are two distinct paths to the goal.



There are two possible paths from cell *(0, 0)* to cell *(1, 3)* in this matrix.

Function Description
Complete the function numberOfPaths in the editor below. The function must return the number of paths through the matrix, modulo (10^9 + 7).
[By doing a modulo, we get around overflow. By doing it with a prime number, we maximize chances of uniform distribution of remainders. By doing it with a large prime like 10^9 + 7, we minimize chances of repeats altogether]

numberOfPaths has the following parameter(s):
2D array of integers a.

Constraints
- *1 ≤ n, m ≤ 1000*
- Each cell in matrix *a* contains either a *0* or a *1*.

Input Format for Custom Testing
Input from stdin will be processed as follows and passed to the function.

The first line contains an integer *n*, the number of rows in matrix *a*.
The next line contains an integer *m*, the number of columns in matrix *a*.
The next *n* lines each contain space separated m integer values, for row a[i] where 0 ≤ i < n.

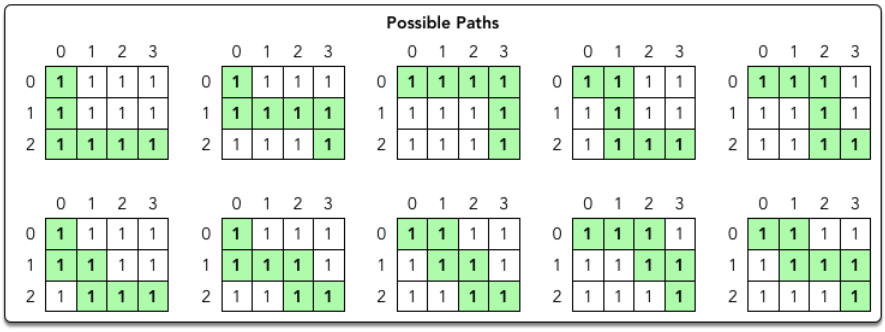
Sample Case 0
Sample Input 0

```
3
4
1 1 1 1
1 1 1 1
1 1 1 1
```

Sample Output 0

```
10
```

Explanation 0



There are *10* possible paths from cell *(0, 0)* to cell *(2, 3)*.

Sample Case 1

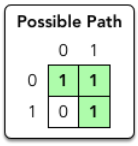
Sample Input 1

```
2
2
1 1
0 1
```

Sample Output 1

```
1
```

Explanation 1



There is *1* possible path from cell *(0, 0)* to cell *(1, 1)*.

Code Editor

Python 3 (python 3.6.6) Theme: Light Reset My Code Auto Complete On Auto Complete Off

Show Input/Output Code

```
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
```

```
def numberOfPaths(a):
    rows = len(a)
    cols = len(a[0])
    dp = [[0] * (cols + 1) for _ in range(rows + 1)]


    for r in range(rows - 1, -1, -1):
        for c in range(cols - 1, -1, -1):
            if r==rows - 1 and c==cols - 1:
                dp[rows - 1][cols - 1] = 1
            elif a[r][c]==0:
                dp[r][c]=0
            else:
                dp[r][c] = dp[r + 1][c] + dp[r][c + 1]

    return dp[0][0] % (10**9+7)
```

☐ **Run against custom input** Quick Test (Takes ~15s) Full Test (Takes ~45s)

Quick Test Result

No result to show!

Full Test Result

No result to show!

```
#Uses half of the memory

def numberOfPaths(a):
    m = len(a)
    n = len(a[0])
    table = [[0 for _ in range(n)] for _ in range(m)]
    table[-1][-1] = a[-1][-1]
    #Fill up last row and last column
    for i in range(m-2, -1, -1):
        if a[i][-1] == 0:
            table[i][-1] = 0
        else:
            table[i][-1] = table[i+1][-1]
    for i in range(n-2, -1, -1):
        if a[-1][i] == 0:
            table[-1][i] = 0
        else:
            table[-1][i] = table[-1][i+1]
    #Fill up rest of table
    for i in range(m-2, -1, -1):
        for j in range(n-2, -1, -1):
            if a[i][j] == 0:
                table[i][j] = 0
            else:
                table[i][j] = table[i][j+1] + table[i+1][j]
    return (table[0][0] % (pow(10,9) + 7))
'''
```