576. Out of Boundary Paths

Add to List

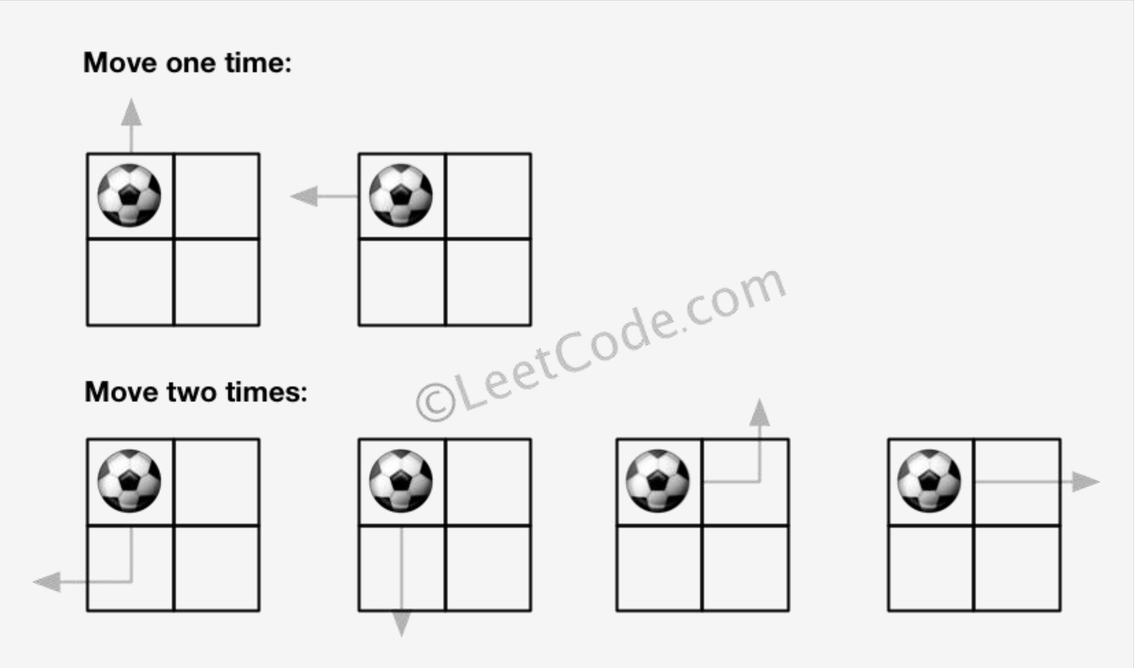
DescriptionHintsSubmissionsSolutions

- Total Accepted: **1539**
- Total Submissions: **4572**
- Difficulty: **Hard**
- Contributors:fallcreek

There is an **m** by **n** grid with a ball. Given the start coordinate **(i,j)** of the ball, you can move the ball to **adjacent** cell or cross the grid boundary in four directions (up, down, left, right). However, you can **at most** move **N** times. Find out the number of paths to move the ball out of grid boundary. The answer may be very large, return it after mod 109 + 7.

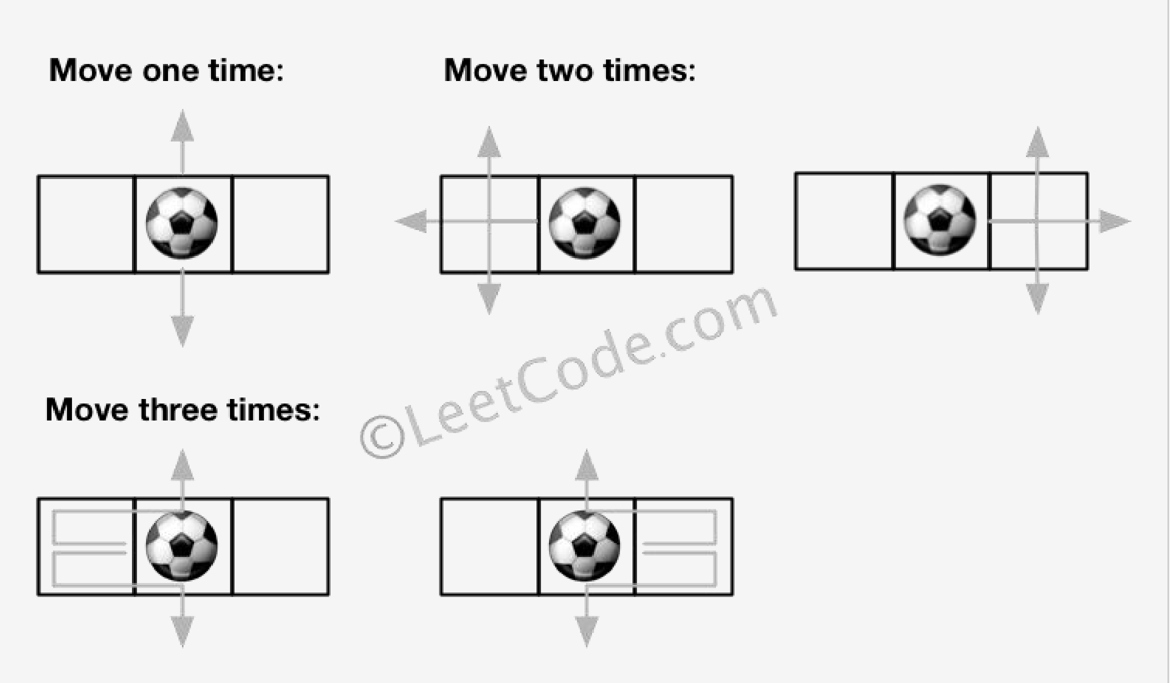
Example 1:

Input:m = 2, n = 2, N = 2, i = 0, j = 0**Output:** 6**Explanation:**



Example 2:

Input:m = 1, n = 3, N = 3, i = 0, j = 1**Output:** 12**Explanation:**



Note:

1. Once you move the ball out of boundary, you cannot move it back.

2. The length and height of the grid is in range [1,50].

3. N is in range [0,50].

Hint DP[m*n][N] means the solution of i*n+j in N steps the answer came from 4 paths: top, down, left, right

```
class Solution {
private:
    int m, n, N, dx[4] = {-1, 1, 0, 0}, dy[4] = {0, 0, -1, 1};
    const int mod = 1e9 + 7;
    bool check(int i, int j) {return i >= m || j >= n || i < 0 || j < 0;}
public:
    int findPaths(int m, int n, int N, int i, int j) {
        this->m = m, this->n = n, this->N = N;
        vector<vector<int>> dp(m * n, vector<int>(N, -1));
        return solve(i, j, N, dp, 0);
    }
    int solve(int i, int j, int step, vector<vector<int>>& dp, int ans) {
        if (check(i, j)) return 1; // out of boundary, count as 1 way
        if (step == 0) return 0; // without steps but not out of bounday,
don't count as a way
        if (dp[i*n + j][step-1] == -1) {
            // the answer came from 4 paths: top, down, left, right
            for (int k=0; k<4; ++k)
                ans = (ans + solve(i + dx[k], j + dy[k], step-1, dp, 0) % mod) %
mod;
            dp[i*n + j][step-1] = ans;
        }
        return dp[i*n + j][step-1];
    }
};
```

Python version:

```
class Solution(object):
    dx = [-1,1,0,0]
    dy = [0,0,-1,1]
    lc = 1e9 + 7

    def solve(self, i, j, step, dp, ans, m, n, N):
        if i >= m or j >= n or i < 0 or j < 0:
            return 1
        if step == 0:
            return 0
        if dp[i*n +j][step - 1] == -1:
            for k in xrange(4):
                ans = (ans + self.solve(i + self.dx[k], j + self.dy[k], step-1,
dp, 0, m, n, N) % self.lc) % self.lc
            dp[i*n + j][step - 1] = ans
        return int(dp[i*n + j][step - 1])

    def findPaths(self, m, n, N, i, j):
        dp = [[-1 for t in xrange(N)] for k in xrange(m*n)]
        return self.solve(i, j, N, dp, 0, m, n, N)
```