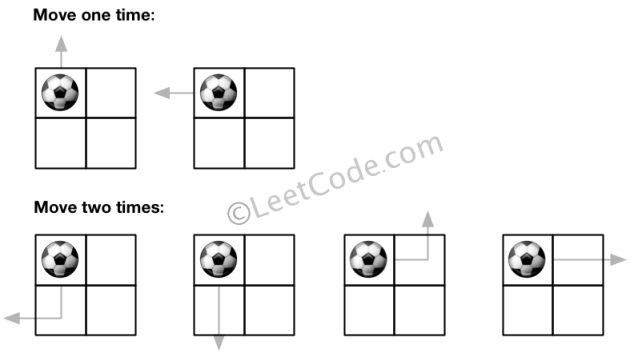
给定一个 **m × n** 的网格和一个球。球的起始坐标为 **(i,j)**，你可以将球移到**相邻**的单元格内，或者往上、下、左、右四个方向上移动使球穿过网格边界。但是，你**最多**可以移动 **N** 次。找出可以将球移出边界的路径数量。答案可能非常大，返回 结果 mod $10^9 + 7$ 的值。

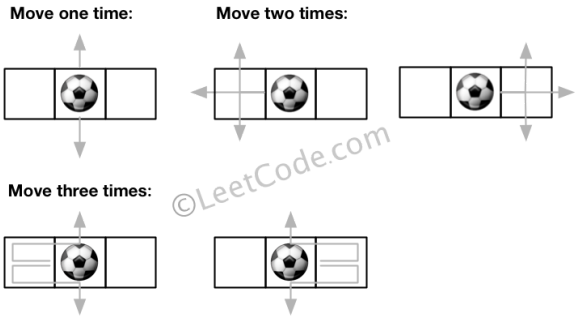
示例 1：

输入： `m = 2, n = 2, N = 2, i = 0, j = 0`
输出： `6`
解释：



示例 2：

输入： `m = 1, n = 3, N = 3, i = 0, j = 1`
输出： `12`
解释：



说明:

1. 球一旦出界，就不能再被移动回网格内。
2. 网格的长度和高度在 [1,50] 的范围内。
3. N 在 [0,50] 的范围内。

分析，路径的话，是有长度一说，移动一步，移动2步，移动3步，第3步能过来的路径数量，等于他第二部四周的路径的和，然后每步能出界的地方为每个路径边界。动态的统计每步能到该格子的路径的个数，并将边界的相加（这个为能出界的路径），需注意传入为0步

时直接返回0，还有路径值计算，需要把加法分开取模，否则会答案不准确。还需记得1e9+7为double，需强转为int

```
class Solution {
public:
    int findPaths(int m, int n, int N, int i, int j) {
        if(N==0){
            return 0;
        }
        vector<vector<int>> nums;
        int result=0;
        for(int k=0;k<m+2;k++){
            vector<int> num(n+2,0);
            nums.push_back(num);
        }
        nums[i+1][j+1]=1;
        for(int k=1;k<m+1;k++){
            result=(result+nums[k][1]+nums[k][n])%(int)(1e9+7);
        }
        for(int k=1;k<n+1;k++){
            result=(result+nums[1][k]+nums[m][k])%(int)(1e9+7);
        }
        for(int l=1;l<N;l++){
            vector<vector<int>> nums1(nums);
            for(int m1=1;m1<m+1;m1++){
                for(int n1=1;n1<n+1;n1++){
                    nums[m1][n1]=(nums1[m1][n1-1]+nums1[m1][n1+1])%(int)(1e9+7);
                    nums[m1][n1]=(nums[m1][n1]+nums1[m1-1][n1])%(int)(1e9+7);
                    nums[m1][n1]=(nums[m1][n1]+nums1[m1+1][n1])%(int)(1e9+7);
                }
            }
            for(int k=1;k<m+1;k++){
            result=(result+nums[k][1])%(int)(1e9+7);
            result=(result+nums[k][n])%(int)(1e9+7);
            }
```

```
            for(int k=1;k<n+1;k++){
            result=(result+nums[1][k])%(int)(1e9+7);
                result=(result+nums[m][k])%(int)(1e9+7);
            }
        }
        return result;
    }
};
```