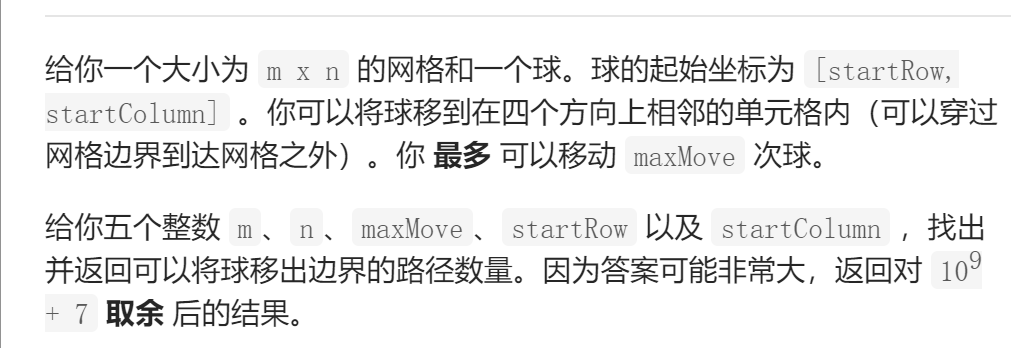
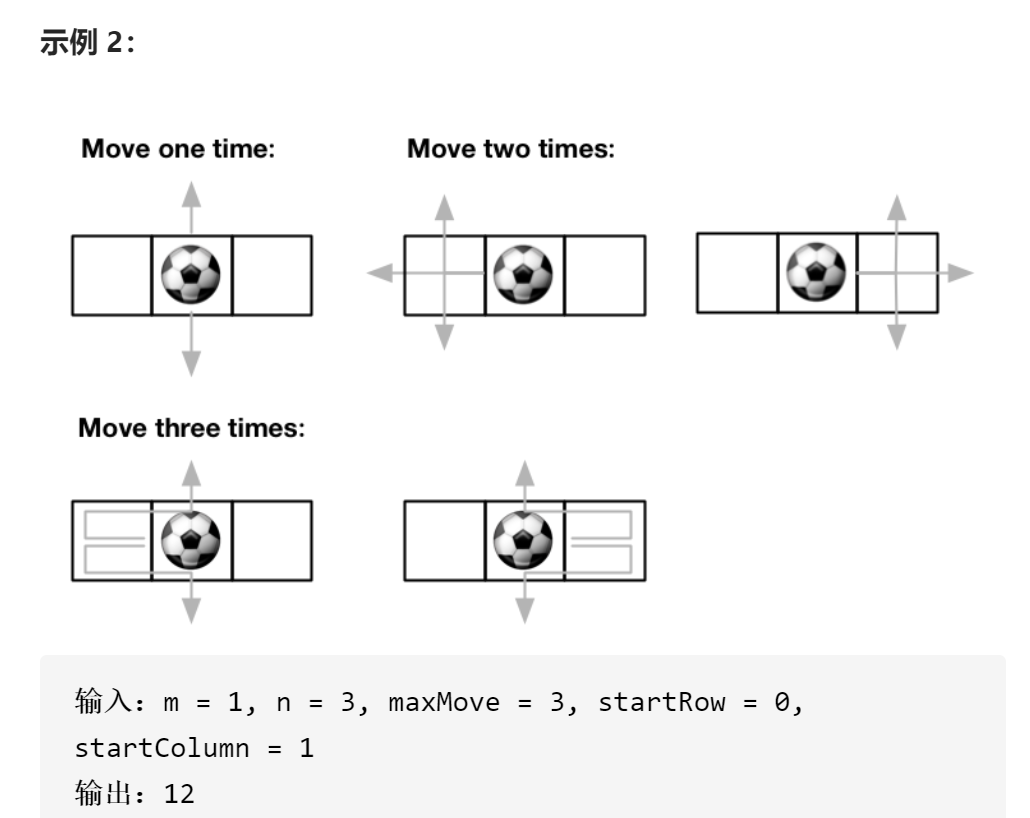
# [576. 出界的路径数](https://leetcode-cn.com/problems/out-of-boundary-paths/)

## 题目描述：





## 深度优先算法超时

class Solution {  
private:  
int res,Maxmove,mod;  
void dfs(int&m,int&n,int steps,int x,int y){  
 if(x<0||y<0||x>=m||y>=n){  
 res%=mod;  
 ++res;  
 return;  
 }  
 if(steps==Maxmove)return;  
 dfs(m,n,steps+1,x+1,y);  
 dfs(m,n,steps+1,x,y+1);  
 dfs(m,n,steps+1,x-1,y);  
 dfs(m,n,steps+1,x,y-1);  
}  
public:  
 int findPaths(int m, int n, int maxMove, int startRow, int startColumn) {  
 //深度优先搜索  
 //深搜超时  
 res=0;  
 Maxmove=maxMove;  
 mod=1000000000+7;  
 dfs(m,n,0,startRow,startColumn);  
 return res;  
  
 }  
};

## 深度优先算法+记忆化搜索

### Dp【i】【j】【k】表示从坐标（i，j)出发，且路径长度为k时满足条件的路径数，以减少重复的搜索

class Solution {  
private:  
int mod,Maxmove;  
vector<vector<vector<int>>>dp;  
// int[][] dirs = new int[][]{{1,0},{-1,0},{0,1},{0,-1}};  
vector<vector<int>>dir;  
  
int dfs(int&m,int&n,int k,int x,int y){  
 if(x<0||y<0||x>=m||y>=n)return 1;  
 if(Maxmove==k)return 0;  
 if(dp[x][y][k]!=-1)return dp[x][y][k];  
 int ans=0;  
 for(int i=0;i<4;i++){  
 ans+=dfs(m,n,k+1,x+dir[i][0],y+dir[i][1]);  
 ans%=mod;  
 }  
 dp[x][y][k]=ans;  
 return ans;  
  
}  
public:  
 int findPaths(int m, int n, int maxMove, int startRow, int startColumn) {  
 //深度优先搜索  
 //深搜超时  
 //改为记忆化搜索--用空间换时间  
 int k=maxMove;  
 Maxmove=maxMove;  
 mod=1000000000+7;  
 dp.assign(m,vector<vector<int>>(n,vector<int>(k,-1)));  
 dir.push\_back({-1,0});  
 dir.push\_back({1,0});  
 dir.push\_back({0,1});  
 dir.push\_back({0,-1});  
  
 return dfs(m,n,0,startRow,startColumn);  
 }  
};