逻辑航线信息学奥赛系列教程

动态规划之路径系列问题

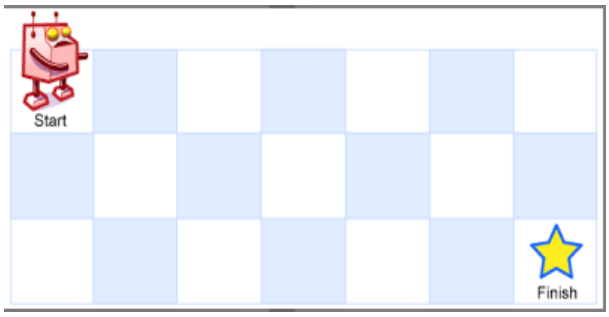
力扣62：不同路径

题目描述

一个机器人位于一个 m x n 网格的左上角 （起始点在下图中标记为 “Start” ）。

机器人每次只能向下或者向右移动一步。机器人试图达到网格的右下角（在下图中标记为 “Finish” ）。

问总共有多少条不同的路径？



示例1：

输入：m = 3, n = 7
输出：28

示例2：

输入：m = 3, n = 2
输出：3
解释：
从左上角开始，总共有 3 条路径可以到达右下角。
1. 向右 -> 向下 -> 向下
2. 向下 -> 向下 -> 向右
3. 向下 -> 向右 -> 向下

解析

这种类型的题，我们之前在讲回溯算法的时候已经非常熟悉了。现在，我们尝试使用动态规划来进行求解。

依然按照动规的五部曲进行解析。

1、确定dp数组及其下标的含义。
设i为行号，j为列号，则有dp[i][j]代表从(0,0)点出发，到达(i,j)总共有几条线路。

2、推导状态转移方程。
因为我们只能向右和向下走，因此dp[i][j]只能来源于自己的左边和上边，即：
dp[i][j]=dp[i][j-1]+dp[i-1][j]

3、初始化DP数组。
第一行和第一列都只有一种走法，即来源于自己的左边和上边。因此dp[i][0]=1,dp[0][j]=1。

4、确定遍历顺序。
遍历顺序从前向后。

5、举例推导DP数组。
以例1为例，开始进行推导，可以得出下图：

	0	1	2	3	4	5	6
0	1	1	1	1	1	1	1
1	1	2	3	4	5	6	7
2	1	3	6	10	15	21	28

编码

```
#include <bits/stdc++.h>

using namespace std;

class Solution {

public:
    //定义动态规划表
    int dp[101][101];

    int uniquePaths(int m, int n) {
        //初始化行列值
        for (int i = 0; i < m; ++i) {
            dp[i][0] = 1;
        }
        for (int j = 0; j < n; ++j) {
            dp[0][j] = 1;
        }
        for (int i = 1; i < m; ++i) {
            for (int j = 1; j < n; ++j) {
                //状态转移
                dp[i][j] = dp[i - 1][j] + dp[i][j - 1];
            }
        }
        //答案存储在最后一个格子中
```

```
            return dp[m - 1][n - 1];
        }
    };


    int main() {
        Solution s{};
        int m, n;
        cin >> m >> n;
        cout << s.uniquePaths(m, n);
        return 0;
    }
```

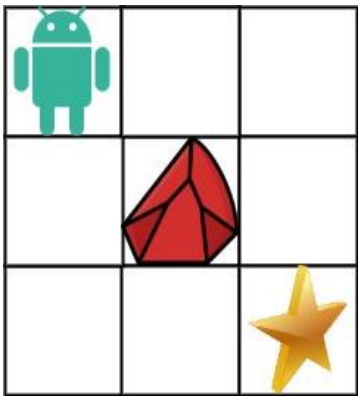
力扣63：不同路径（Ⅱ）

题目描述

一个机器人位于一个 m x n 网格的左上角 （起始点在下图中标记为“Start” ）。

机器人每次只能向下或者向右移动一步。机器人试图达到网格的右下角（在下图中标记为“Finish”）。

现在考虑网格中有障碍物。那么从左上角到右下角将会有多少条不同的路径？



示例1：

输入：obstacleGrid = [[0,0,0],[0,1,0],[0,0,0]]
输出：2
解释：
3x3 网格的正中间有一个障碍物。
从左上角到右下角一共有 2 条不同的路径：
1. 向右 -> 向右 -> 向下 -> 向下
2. 向下 -> 向下 -> 向右 -> 向右

解析

本题比上一题增加了一些难度，即存在了障碍物。很明显，这些障碍物是不可通行的，因此存在障碍物的地点的可通行方法数即为0.

我们依然按照五部曲进行解析。

1、确定dp数组及其下标的含义。
设i为行号，j为列号，则有dp[i][j]代表从(0,0)点出发，到达(i,j)总共有几条线路。

2、推导状态转移方程。
因为我们只能向右和向下走，因此dp[i][j]只能来源于自己的左边和上边，即：
dp[i][j]=dp[i][j-1]+dp[i-1][j]

3、初始化DP数组。
第一行和第一列都只有一种走法，即来源于自己的左边和上边。但是要注意，这里可能存在障碍物，因此我们必须要判断obstacleGrid[i][0]和obstacleGrid[0][j]是否可以通行，然后再对dp[i][0]和dp[0][j]进行赋值。

4、确定遍历顺序。
遍历顺序从前向后。

5、举例推导DP数组。
以例1为例，开始进行推导，可以得出下图：

	0	1	2
0	1	1	1
1	1	0	1
2	1	1	2

中间存在障碍，因此无法通行。

编码

```
#include <bits/stdc++.h>

using namespace std;

class Solution {

public:
    //定义动态规划表
    int dp[101][101];

    int uniquePathsWithObstacles(vector<vector<int>> &obstacleGrid) {
        //行数
        int m = obstacleGrid.size();
        //列数
        int n = obstacleGrid[0].size();
        //初始化行列值,如果出现了障碍，则停止赋值
        //形如：1 1 1 1 1 0 0 0 0
        for (int i = 0; i < m && obstacleGrid[i][0] == 0; ++i) {
```

```
            dp[i][0] = 1;
        }
        //同上
        for (int j = 0; j < n && obstacleGrid[0][j] == 0; ++j) {
            dp[0][j] = 1;
        }
        for (int i = 1; i < m; ++i) {
            for (int j = 1; j < n; ++j) {
                //判断是否存在障碍
                if (obstacleGrid[i][j] != 1) {
                    //状态转移
                    dp[i][j] = dp[i - 1][j] + dp[i][j - 1];
                }
            }
        }
        //答案存储在最后一个格子中
        return dp[m - 1][n - 1];
    }
};


int main() {
    Solution s{};
    vector<vector<int>> obstacleGrid = {{0, 0, 0},
                                        {0, 1, 0},
                                        {0, 0, 0}};
    cout << s.uniquePathsWithObstacles(obstacleGrid);
    return 0;
}
```