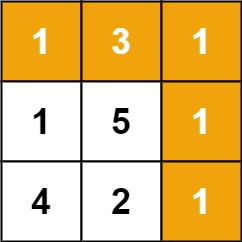
# 题目

给定一个包含非负整数的 m×n 网格grid，请找出一条从左上角到右下角的路径，使得路径上的数字总和为最小。

说明：每次只能向下或者向右移动一步。

示例 1：



输入：grid = [[1,3,1],[1,5,1],[4,2,1]]

输出：7

解释：因为路径 1→3→1→1→1 的总和最小。

示例 2：

输入：grid = [[1,2,3],[4,5,6]]

输出：12

提示：

m == grid.length

n == grid[i].length

1 <= m, n <= 200

0 <= grid[i][j] <= 100

# 分析

## 方法一：动态规划

**思路：**

**代码：**

public:

// 动态规划：从起始点到终点

int minPathSum(vector<vector<int>>& grid) {

int m = grid.size(), n = grid[0].size();

// 状态定义：dp[i][j]表示从[0,0]到[i,j]的最小路径和

vector<vector<int>> dp(m, vector<int>(n));

// 状态初始化

dp[0][0] = grid[0][0];

// 状态转移

for (int i = 0; i < m; i++) {

for (int j = 0; j < n; j++) {

if (i == 0 && j != 0) { //第一行

dp[i][j] = grid[i][j] + dp[i][j - 1];

} else if (i != 0 && j == 0) { //第一列

dp[i][j] = grid[i][j] + dp[i - 1][j];

} else if (i != 0 && j != 0) { //这个条件不要漏掉

dp[i][j] = grid[i][j] + min(dp[i - 1][j], dp[i][j - 1]);

}

}

}

// 返回结果

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

}

}