Leetcode\_64\_MinimumPathSum\_最小路径和\_Medium

# Leetcode\_64\_MinimumPathSum\_最小路径和\_Medium

## 题目介绍

\* 难度：Medium 实际是：Easy

\* <https://leetcode.com/problems/minimum-path-sum/description/>

\* 题目介绍：

\* Given a m x n grid filled with non-negative numbers,

\* find a path from top left to bottom right

\* which minimizes the sum of all numbers along its path.

\* Note: You can only move either down or right at any point in time.

\* Example:

\* Input:

\* [

\* [1,3,1],

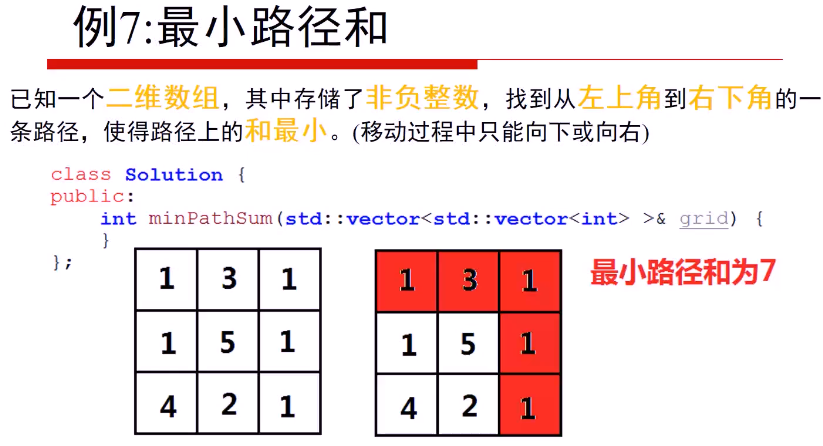
\* [1,5,1],

\* [4,2,1]

\* ]

\* Output: 7

\* Explanation: Because the path 1→3→1→1→1 minimizes the sum.



## 思路分析

\* 思路分析：动态规划状态转移方程：

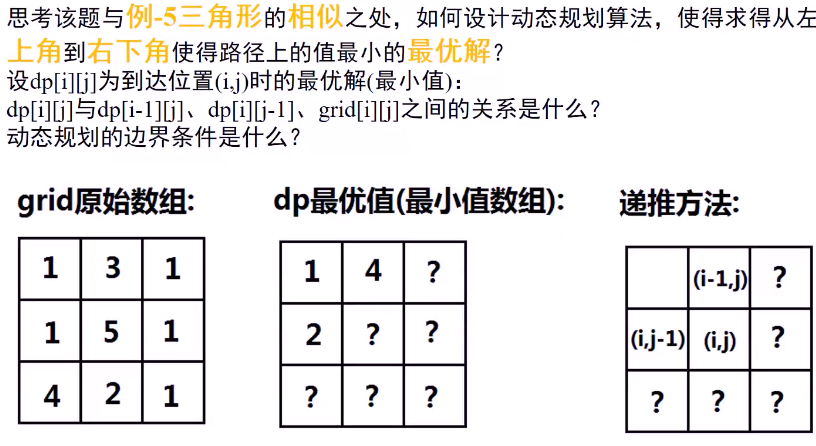
**\* dp[i][j] = min{dp[i][j-1],dp[i-1][j]} + grid[i][j];其中，i,j>=1**

**\* 边界条件：**

**\* dp[0][0] = grid[0][0];**

**\* dp[0][j] = dp[0][j-1] + grid[0][j];**

**\* dp[i][0] = dp[i-1][0] + grid[i][0];**



## Java代码

public int **minPathSum**(int[][] grid) {

if(grid == null||grid.length == 0) return 0;

int rows = grid.length,columns = grid[0].length;

**int[][] dp = new int[rows][columns];**

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

//初始化第一行

for(int j = 1;j < columns;j++)

**dp[0][j] = dp[0][j-1] + grid[0][j];**

//初始化第一列

for(int i = 1;i < rows;i++)

**dp[i][0] = dp[i-1][0] + grid[i][0];**

//遍历i，j，求dp[i][j]

for(int i = 1;i < rows;i++){

for(int j = 1;j < columns;j++){

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

}

}

return dp[rows-1][columns-1];//返回终点位置的dp值

}

