

*//这题呢也是很明显的动态规划问题，局部最优解的累加至全局最优解*

*//创建一个二维数组[i,j], 数组中每一项的值记录着到达当前点的最优解*

*//思路：由于移动机制问题，第一行和第一列的各自的最优解只能是其左边的累加和上边的累加，*

*// 然后*

*//先定义一个二维数组m*

const gird = [

    [1,3,1],

    [1,5,1],

    [4,2,1]

  ]

function maxValue(gird) {

    const m = gird.length;

    const n = gird[0].length

    dp = [];

    for(let i = 0; i < m; i ++){

        dp.push(new Array(n).fill(0))

    }

*//先把边界的固定值给确定好*

    dp[0][0] = gird[0][0]

    for(let i = 1; i < n; i++) {

        dp[0][i] = dp[0][i-1] + gird[0][i]

    }

    for(let j = 1; j < m; j++) {

        dp[j][0] = dp[j-1][0] + gird[j][0]

    }

*//通过边界来慢慢构建内部的最佳值*

    for(let i = 1; i < m; i++) {

        for(let j = 1; j < n; j++) {

            dp[i][j] = dp[i-1][j] > dp[i][j-1] ? dp[i-1][j] + gird[i][j] : dp[i][j-1] + gird[i][j]

        }

    }

    return dp[m-1][n-1]

}

console.log(maxValue(gird))