```swift
import Foundation
/// 剑指 Offer 47. 礼物的最大价值
/// 在一个 `m*n` 的棋盘的每一格都放有一个礼物，每个礼物都有一定的价值（价值大于 0）。
/// 你可以从棋盘的左上角开始拿格子里的礼物，并每次向右或者向下移动一格、
/// 直到到达棋盘的右下角。给定一个棋盘及其上面的礼物的价值，
/// 请计算你最多能拿到多少价值的礼物?
///
/// 示例 1:
/// 输入:
/// `[[1,3,1],`
///  `[1,5,1],`
///  `[4,2,1]]`
/// 输出: 12
/// 解释: 路径 1→3→5→2→1 可以拿到最多价值的礼物
class Solution {
    /// 设动态规划矩阵`dp`
    /// `dp(i,j)`代表从棋盘的左上角开始，到达单元格时能拿到礼物的最大累计价值。
    /// 每一个格子的最优解是由左边与上面两个格子的最优解来推导而来
    func maxValue(_ grid: [[Int]]) -> Int {
        var _grid: [[Int]] = grid
        for i in 0..<_grid.count {
            for j in 0..<_grid[0].count {
                /// 第一个格子不做处理
                if (i == 0) && (j == 0) { continue }
                if i == 0 {
                    /// 第一行的格子只能从左边的格子获得到
                    _grid[i][j] += _grid[i][j-1]
                } else if j == 0 {
                    /// 第一列的格子只能从上面的格子获得到
                    _grid[i][j] += _grid[i-1][j]
                } else {
                    /// 当前格子的数值 = max[左边的格子，上面的格子] + grid当前的格子
                    _grid[i][j] += max(_grid[i-1][j], _grid[i][j-1])
                }
            }
        }
        /// 最右下角的数即为最大值
        return _grid[_grid.count-1][_grid[0].count-1]
    }
}

let solution = Solution()
let grid = [[1,3,1],
            [1,5,1],
            [4,2,1]]
print(solution.maxValue(grid))
```