# 三维形体的表面积

给你一个 `n * n` 的网格 `grid` ，上面放置着一些 `1 x 1 x 1` 的正方体。

每个值 `v = grid[i][j]` 表示 `v` 个正方体叠放在对应单元格 `(i, j)` 上。

放置好正方体后，任何直接相邻的正方体都会互相粘在一起，形成一些不规则的三维形体。

请你返回最终这些形体的总表面积。

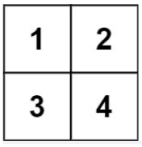
**注意：**每个形体的底面也需要计入表面积中。

**示例 1：**


**输入：**`grid = [[2]]`

**输出：**`10`

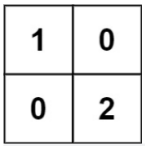
**示例 2：**




**输入：**`grid = [[1,2],[3,4]]`

**输出：**`34`

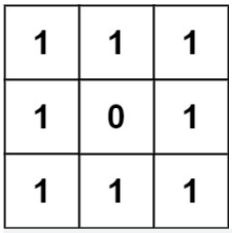
**示例 3：**




**输入：**`grid = [[1,0],[0,2]]`

**输出：**`16`

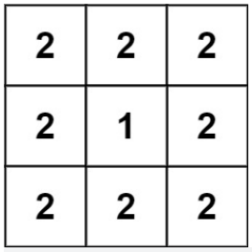
**示例 4：**



输入：grid = [[1,1,1],[1,0,1],[1,1,1]]

输出：32

示例 5：



输入：grid = [[2,2,2],[2,1,2],[2,2,2]]

输出：46

```
class Solution:
    def surfaceArea(self, grid: List[List[int]]) -> int:
        if not grid:
            return 0
        blocks=0
        covers=0
        for i in range(len(grid)):
            for j in range(len(grid[0])):
                blocks+=grid[i][j]
                if grid[i][j]>1:
                    covers+=grid[i][j]-1
                else:
                    covers+=0
                if i>0:
                    covers+=min(grid[i-1][j],grid[i][j])#同一列找最矮的计算重
叠
                if j>0:
                    covers+=min(grid[i][j-1],grid[i][j])#同一行找最矮的计算重
叠
        return blocks*6-covers*2 #每一个立方体都有 6 个面 重合一次减去 2 个面
```