在二维数组grid中，grid[i][j]代表位于某处的建筑物的高度。 我们被允许增加任何数量（不同建筑物的数量可能不同）的建筑物的高度。 高度 0 也被认为是建筑物。
最后，从新数组的所有四个方向（即顶部，底部，左侧和右侧）观看的“天际线”必须与原始数组的天际线相同。 城市的天际线是从远处观看时，由所有建筑物形成的矩形的外部轮廓。 请看下面的例子。
建筑物高度可以增加的最大总和是多少?

例子:
输入: grid = [[3,0,8,4],[2,4,5,7],[9,2,6,3],[0,3,1,0]]
输出: 35
解释:

```
The grid is:
[ [3, 0, 8, 4],
  [2, 4, 5, 7],
  [9, 2, 6, 3],
  [0, 3, 1, 0] ]

从数组竖直方向（即顶部，底部）看“天际线”是: [9, 4, 8, 7]
从水平水平方向（即左侧，右侧）看“天际线”是: [8, 7, 9, 3]

在不影响天际线的情况下对建筑物进行增高后，新数组如下:

gridNew = [ [8, 4, 8, 7],
            [7, 4, 7, 7],
            [9, 4, 8, 7],
            [3, 3, 3, 3] ]
```

说明:

- 1 < grid.length = grid[0].length <= 50。
- grid[i][j] 的高度范围是: [0, 100]。
- 一座建筑物占据一个grid[i][j]：换言之，它们是 1 x 1 x grid[i][j] 的长方体。

思路：动态的进行计算，统计行最大值和列最大值，一个建筑能增大的最大值为其对应行最大值和列最大值中的最小值。统计完后再进行每个建筑进行最大新增的判断。

代码

```
class Solution {
public:
```

```
    int maxIncreaseKeepingSkyline(vector<vector<int>>& grid) {
        vector<int> row;
        vector<int> line;
        for(int i=0;i<grid.size();i++){
            int max=grid[i][0];
            for(int j=1;j<grid[i].size();j++){
                if(grid[i][j]>max){
                    max=grid[i][j];
                }
            }
            row.push_back(max);
        }
        for(int i=0;i<grid[0].size();i++){
            int max=grid[0][i];
            for(int j=1;j<grid.size();j++){
                if(grid[j][i]>max){
                    max=grid[j][i];
                }
            }
            line.push_back(max);
        }
        int num=0;
        for(int i=0;i<grid.size();i++){
            for(int j=0;j<grid[i].size();j++){
                int k=row[i]<line[j]?row[i]:line[j];
                num=num+k-grid[i][j];
            }
        }
        return num;
    }
};
```