807. Max Increase to Keep City Skyline

Description (/problems/max-increase-to-keep-city-skyline/description/) Hints (/problems/max-increase-to-keep-city-skyline/hir

Pick One (/problems/random-one-question/)

In a 2 dimensional array `grid`, each value `grid[i][j]` represents the height of a building located there. We are allowed to increase the height of any number of buildings, by any amount (the amounts can be different for different buildings). Height 0 is considered to be a building as well.

At the end, the "skyline" when viewed from all four directions of the grid, i.e. top, bottom, left, and right, must be the same as the skyline of the original grid. A city's skyline is the outer contour of the rectangles formed by all the buildings when viewed from a distance. See the following example.

What is the maximum total sum that the height of the buildings can be increased?

```
Example:
Input: grid = [[3,0,8,4],[2,4,5,7],[9,2,6,3],[0,3,1,0]]
Output: 35
Explanation:
The grid is:
[ [3, 0, 8, 4],
  [2, 4, 5, 7],
  [9, 2, 6, 3],
  [0, 3, 1, 0] ]

The skyline viewed from top or bottom is: [9, 4, 8, 7]
The skyline viewed from left or right is: [8, 7, 9, 3]

The grid after increasing the height of buildings without affecting skylines is:

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

Notes:

- `1 < grid.length = grid[0].length <= 50`.
- All heights `grid[i][j]` are in the range `[0, 100]`.
- All buildings in `grid[i][j]` occupy the entire grid cell: that is, they are a `1 x 1 x grid[i][j]` rectangular prism.

Seen this question in a real interview before? Yes No

Companies

Google 3 (/company/google)

Java

```java
class Solution {
    public int maxIncreaseKeepingSkyline(int[][] grid) {
        if (grid.length == 0 || grid[0].length == 0) {
            return 0;
        }
        int[] maxLeft = new int[grid.length];
        int[] maxTop = new int[grid[0].length];

        for (int i = 0; i < grid.length; i++) {
            for (int j=0; j<grid[i].length; j++) {
                if (grid[i][j] > maxLeft[i]) {
                    maxLeft[i] = grid[i][j];
                }
                if (grid[i][j] > maxTop[j]) {
                    maxTop[j] = grid[i][j];
                }
            }
        }

        int sum = 0;
        for (int i = 0; i < grid.length; i++) {
            for (int j=0; j<grid[i].length; j++) {
                sum += Math.min(maxLeft[i], maxTop[j]) - grid[i][j];
            }
        }
        return sum;
    }
}
```

Notes

Custom Testcase (**Contribute**)

Run Submit