Problem 807: Max Increase to Keep City Skyline

Problem Information

Difficulty: Medium
Acceptance Rate: 0.00%
Paid Only: No

Problem Description

There is a city composed of

n x n

blocks, where each block contains a single building shaped like a vertical square prism. You are given a

0-indexed

n x n

integer matrix

grid

where

grid[r][c]

represents the

height

of the building located in the block at row

r

and column

c

.

A city's

skyline

is the outer contour formed by all the building when viewing the side of the city from a distance. The

skyline

from each cardinal direction north, east, south, and west may be different.

We are allowed to increase the height of

any number of buildings by any amount

(the amount can be different per building). The height of a

0

-height building can also be increased. However, increasing the height of a building should

not

affect the city's

skyline

from any cardinal direction.

Return

the

maximum total sum

that the height of the buildings can be increased by

without

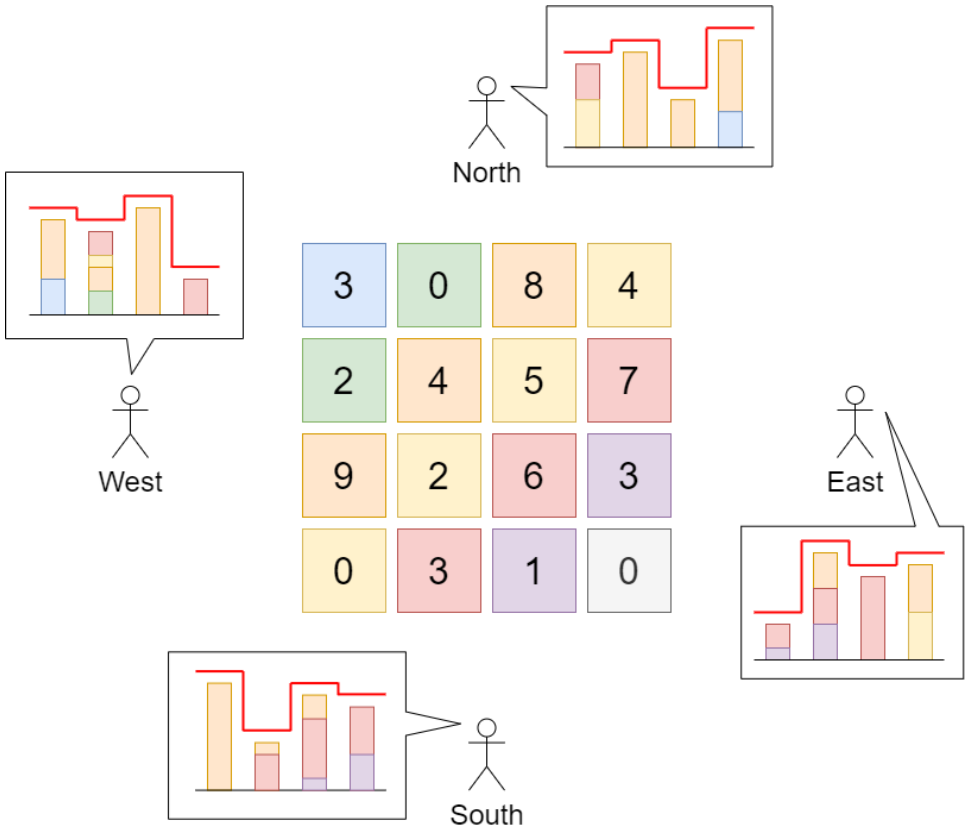
changing the city's

skyline

from any cardinal direction

.

Example 1:



Input:

grid = [[3,0,8,4],[2,4,5,7],[9,2,6,3],[0,3,1,0]]

Output:

35

Explanation:

The building heights are shown in the center of the above image. The skylines when viewed from each cardinal direction are drawn in red. 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]]

Example 2:

Input:

grid = [[0,0,0],[0,0,0],[0,0,0]]

Output:

0

Explanation:

Increasing the height of any building will result in the skyline changing.

Constraints:

n == grid.length

n == grid[r].length

2 <= n <= 50

0 <= grid[r][c] <= 100

Code Snippets

C++:

```
class Solution {
public:
int maxIncreaseKeepingSkyline(vector<vector<int>>& grid) {

}
};
```

Java:

```
class Solution {
public int maxIncreaseKeepingSkyline(int[][] grid) {

}
}
```

Python3:

```
class Solution:
def maxIncreaseKeepingSkyline(self, grid: List[List[int]]) -> int:
```

Python:

```
class Solution(object):
def maxIncreaseKeepingSkyline(self, grid):
"""
:type grid: List[List[int]]
:rtype: int
"""
```

JavaScript:

```
/**
* @param {number[][]} grid
* @return {number}
*/
var maxIncreaseKeepingSkyline = function(grid) {

};
```

TypeScript:

```
function maxIncreaseKeepingSkyline(grid: number[][]): number {
```

```
};
```

C#:

```
public class Solution {
public int MaxIncreaseKeepingSkyline(int[][] grid) {

}
}
```

C:

```
int maxIncreaseKeepingSkyline(int** grid, int gridSize, int* gridColSize) {

}
```

Go:

```
func maxIncreaseKeepingSkyline(grid [][]int) int {

}
```

Kotlin:

```
class Solution {
fun maxIncreaseKeepingSkyline(grid: Array<IntArray>): Int {

}
}
```

Swift:

```
class Solution {
func maxIncreaseKeepingSkyline(_ grid: [[Int]]) -> Int {

}
}
```

Rust:

```
impl Solution {
pub fn max_increase_keeping_skyline(grid: Vec<Vec<i32>>) -> i32 {
```

```
}
}
```

Ruby:

```
# @param {Integer[][]} grid
# @return {Integer}
def max_increase_keeping_skyline(grid)

end
```

PHP:

```
class Solution {

/**
* @param Integer[][] $grid
* @return Integer
*/
function maxIncreaseKeepingSkyline($grid) {

}
}
```

Dart:

```
class Solution {
int maxIncreaseKeepingSkyline(List<List<int>> grid) {

}
}
```

Scala:

```
object Solution {
def maxIncreaseKeepingSkyline(grid: Array[Array[Int]]): Int = {

}
}
```

Elixir:

```
defmodule Solution do
@spec max_increase_keeping_skyline(grid :: [[integer]]) :: integer
def max_increase_keeping_skyline(grid) do

end
end
```

Erlang:

```
-spec max_increase_keeping_skyline(Grid :: [[integer()]]) -> integer().
max_increase_keeping_skyline(Grid) ->
.
```

Racket:

```
(define/contract (max-increase-keeping-skyline grid)
(-> (listof (listof exact-integer?)) exact-integer?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Max Increase to Keep City Skyline
* Difficulty: Medium
* Tags: array, greedy
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

class Solution {
public:
int maxIncreaseKeepingSkyline(vector<vector<int>>& grid) {

}
};
```

Java Solution:

```
/**
* Problem: Max Increase to Keep City Skyline
* Difficulty: Medium
* Tags: array, greedy
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

class Solution {
public int maxIncreaseKeepingSkyline(int[][] grid) {

}
}
```

Python3 Solution:

```
"""
Problem: Max Increase to Keep City Skyline
Difficulty: Medium
Tags: array, greedy

Approach: Use two pointers or sliding window technique
Time Complexity: O(n) or O(n log n)
Space Complexity: O(1) to O(n) depending on approach
"""

class Solution:
def maxIncreaseKeepingSkyline(self, grid: List[List[int]]) -> int:
# TODO: Implement optimized solution
pass
```

Python Solution:

```
class Solution(object):
def maxIncreaseKeepingSkyline(self, grid):
"""
:type grid: List[List[int]]
:rtype: int
"""
```

JavaScript Solution:

```
/**
* Problem: Max Increase to Keep City Skyline
* Difficulty: Medium
* Tags: array, greedy
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

/**
* @param {number[][]} grid
* @return {number}
*/
var maxIncreaseKeepingSkyline = function(grid) {

};
```

TypeScript Solution:

```
/**
* Problem: Max Increase to Keep City Skyline
* Difficulty: Medium
* Tags: array, greedy
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

function maxIncreaseKeepingSkyline(grid: number[][]): number {

};
```

C# Solution:

```
/*
* Problem: Max Increase to Keep City Skyline
* Difficulty: Medium
* Tags: array, greedy
*
```

```
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

public class Solution {
public int MaxIncreaseKeepingSkyline(int[][] grid) {

}
}
```

C Solution:

```
/*
* Problem: Max Increase to Keep City Skyline
* Difficulty: Medium
* Tags: array, greedy
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

int maxIncreaseKeepingSkyline(int** grid, int gridSize, int* gridColSize) {

}
```

Go Solution:

```
// Problem: Max Increase to Keep City Skyline
// Difficulty: Medium
// Tags: array, greedy
//
// Approach: Use two pointers or sliding window technique
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(1) to O(n) depending on approach

func maxIncreaseKeepingSkyline(grid [][]int) int {

}
```

Kotlin Solution:

```
class Solution {
fun maxIncreaseKeepingSkyline(grid: Array<IntArray>): Int {

}
}
```

Swift Solution:

```
class Solution {
func maxIncreaseKeepingSkyline(_ grid: [[Int]]) -> Int {

}
}
```

Rust Solution:

```
// Problem: Max Increase to Keep City Skyline
// Difficulty: Medium
// Tags: array, greedy
//
// Approach: Use two pointers or sliding window technique
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(1) to O(n) depending on approach

impl Solution {
pub fn max_increase_keeping_skyline(grid: Vec<Vec<i32>>) -> i32 {

}
}
```

Ruby Solution:

```
# @param {Integer[][]} grid
# @return {Integer}
def max_increase_keeping_skyline(grid)

end
```

PHP Solution:

```
class Solution {

/**
* @param Integer[][] $grid
* @return Integer
*/
function maxIncreaseKeepingSkyline($grid) {

}
}
```

Dart Solution:

```
class Solution {
int maxIncreaseKeepingSkyline(List<List<int>> grid) {

}
}
```

Scala Solution:

```
object Solution {
def maxIncreaseKeepingSkyline(grid: Array[Array[Int]]): Int = {

}
}
```

Elixir Solution:

```
defmodule Solution do
@spec max_increase_keeping_skyline(grid :: [[integer]]) :: integer
def max_increase_keeping_skyline(grid) do

end
end
```

Erlang Solution:

```
-spec max_increase_keeping_skyline(Grid :: [[integer()]]) -> integer().
max_increase_keeping_skyline(Grid) ->
.
```

Racket Solution:

```
(define/contract (max-increase-keeping-skyline grid)
(-> (listof (listof exact-integer?)) exact-integer?)
)
```