Problem 1183: Maximum Number of Ones

Problem Information

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

Problem Description

Consider a matrix

M

with dimensions

width * height

, such that every cell has value

0

or

1

, and any

square

sub-matrix of

M

of size

sideLength * sideLength

has at most

maxOnes

ones.

Return the maximum possible number of ones that the matrix

M

can have.

Example 1:

Input:

width = 3, height = 3, sideLength = 2, maxOnes = 1

Output:

4

Explanation:

In a 3*3 matrix, no 2*2 sub-matrix can have more than 1 one. The best solution that has 4 ones is: [1,0,1] [0,0,0] [1,0,1]

Example 2:

Input:

width = 3, height = 3, sideLength = 2, maxOnes = 2

Output:

6

Explanation:

[1,0,1] [1,0,1] [1,0,1]

Constraints:

1 <= width, height <= 100

1 <= sideLength <= width, height

0 <= maxOnes <= sideLength * sideLength

Code Snippets

C++:

```
class Solution {
public:
int maximumNumberOfOnes(int width, int height, int sideLength, int maxOnes) {

}
};
```

Java:

```
class Solution {
public int maximumNumberOfOnes(int width, int height, int sideLength, int
maxOnes) {

}
}
```

Python3:

```
class Solution:
def maximumNumberOfOnes(self, width: int, height: int, sideLength: int,
maxOnes: int) -> int:
```

Python:

```
class Solution(object):
def maximumNumberOfOnes(self, width, height, sideLength, maxOnes):
"""
:type width: int
:type height: int
:type sideLength: int
:type maxOnes: int
:rtype: int
"""
```

JavaScript:

```
/**
* @param {number} width
* @param {number} height
* @param {number} sideLength
* @param {number} maxOnes
* @return {number}
*/
var maximumNumberOfOnes = function(width, height, sideLength, maxOnes) {

};
```

TypeScript:

```
function maximumNumberOfOnes(width: number, height: number, sideLength:
number, maxOnes: number): number {

};
```

C#:

```
public class Solution {
public int MaximumNumberOfOnes(int width, int height, int sideLength, int
maxOnes) {

}
}
```

C:

```
int maximumNumberOfOnes(int width, int height, int sideLength, int maxOnes) {

}
```

Go:

```
func maximumNumberOfOnes(width int, height int, sideLength int, maxOnes int)
int {

}
```

Kotlin:

```
class Solution {
fun maximumNumberOfOnes(width: Int, height: Int, sideLength: Int, maxOnes:
Int): Int {

}
}
```

Swift:

```
class Solution {
func maximumNumberOfOnes(_ width: Int, _ height: Int, _ sideLength: Int, _
maxOnes: Int) -> Int {

}
}
```

Rust:

```
impl Solution {
pub fn maximum_number_of_ones(width: i32, height: i32, side_length: i32,
max_ones: i32) -> i32 {

}
}
```

Ruby:

```
# @param {Integer} width
# @param {Integer} height
# @param {Integer} side_length
# @param {Integer} max_ones
# @return {Integer}
def maximum_number_of_ones(width, height, side_length, max_ones)

end
```

PHP:

```
class Solution {

/**
* @param Integer $width
* @param Integer $height
* @param Integer $sideLength
* @param Integer $maxOnes
* @return Integer
*/
function maximumNumberOfOnes($width, $height, $sideLength, $maxOnes) {

}
}
```

Dart:

```
class Solution {
int maximumNumberOfOnes(int width, int height, int sideLength, int maxOnes) {

}
}
```

Scala:

```
object Solution {
def maximumNumberOfOnes(width: Int, height: Int, sideLength: Int, maxOnes:
Int): Int = {

}
}
```

Elixir:

```
defmodule Solution do
@spec maximum_number_of_ones(width :: integer, height :: integer, side_length
:: integer, max_ones :: integer) :: integer
def maximum_number_of_ones(width, height, side_length, max_ones) do

end
end
```

Erlang:

```
-spec maximum_number_of_ones(Width :: integer(), Height :: integer(),
SideLength :: integer(), MaxOnes :: integer()) -> integer().
maximum_number_of_ones(Width, Height, SideLength, MaxOnes) ->
.
```

Racket:

```
(define/contract (maximum-number-of-ones width height sideLength maxOnes)
(-> exact-integer? exact-integer? exact-integer? exact-integer?
exact-integer?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Maximum Number of Ones
* Difficulty: Hard
* Tags: greedy, math, sort, queue, heap
*
* Approach: Greedy algorithm with local optimal choices
* Time Complexity: O(n) to O(n^2) depending on approach
* Space Complexity: O(1) to O(n) depending on approach
*/

class Solution {
public:
int maximumNumberOfOnes(int width, int height, int sideLength, int maxOnes) {

}
};
```

Java Solution:

```
/**
* Problem: Maximum Number of Ones
* Difficulty: Hard
* Tags: greedy, math, sort, queue, heap
*
* Approach: Greedy algorithm with local optimal choices
* Time Complexity: O(n) to O(n^2) depending on approach
* Space Complexity: O(1) to O(n) depending on approach
*/

class Solution {
public int maximumNumberOfOnes(int width, int height, int sideLength, int
maxOnes) {

}
}
```

Python3 Solution:

```
"""
Problem: Maximum Number of Ones
Difficulty: Hard
Tags: greedy, math, sort, queue, heap

Approach: Greedy algorithm with local optimal choices
Time Complexity: O(n) to O(n^2) depending on approach
Space Complexity: O(1) to O(n) depending on approach
"""

class Solution:
def maximumNumberOfOnes(self, width: int, height: int, sideLength: int,
maxOnes: int) -> int:
# TODO: Implement optimized solution
pass
```

Python Solution:

```
class Solution(object):
def maximumNumberOfOnes(self, width, height, sideLength, maxOnes):
"""
:type width: int
```

```
:type height: int
:type sideLength: int
:type maxOnes: int
:rtype: int
"""
```

JavaScript Solution:

```
/**
* Problem: Maximum Number of Ones
* Difficulty: Hard
* Tags: greedy, math, sort, queue, heap
*
* Approach: Greedy algorithm with local optimal choices
* Time Complexity: O(n) to O(n^2) depending on approach
* Space Complexity: O(1) to O(n) depending on approach
*/

/**
* @param {number} width
* @param {number} height
* @param {number} sideLength
* @param {number} maxOnes
* @return {number}
*/
var maximumNumberOfOnes = function(width, height, sideLength, maxOnes) {

};
```

TypeScript Solution:

```
/**
* Problem: Maximum Number of Ones
* Difficulty: Hard
* Tags: greedy, math, sort, queue, heap
*
* Approach: Greedy algorithm with local optimal choices
* Time Complexity: O(n) to O(n^2) depending on approach
* Space Complexity: O(1) to O(n) depending on approach
*/
```

```
function maximumNumberOfOnes(width: number, height: number, sideLength:
number, maxOnes: number): number {

};
```

C# Solution:

```
/*
* Problem: Maximum Number of Ones
* Difficulty: Hard
* Tags: greedy, math, sort, queue, heap
*
* Approach: Greedy algorithm with local optimal choices
* Time Complexity: O(n) to O(n^2) depending on approach
* Space Complexity: O(1) to O(n) depending on approach
*/

public class Solution {
public int MaximumNumberOfOnes(int width, int height, int sideLength, int
maxOnes) {

}
}
```

C Solution:

```
/*
* Problem: Maximum Number of Ones
* Difficulty: Hard
* Tags: greedy, math, sort, queue, heap
*
* Approach: Greedy algorithm with local optimal choices
* Time Complexity: O(n) to O(n^2) depending on approach
* Space Complexity: O(1) to O(n) depending on approach
*/

int maximumNumberOfOnes(int width, int height, int sideLength, int maxOnes) {

}
```

Go Solution:

```
// Problem: Maximum Number of Ones
// Difficulty: Hard
// Tags: greedy, math, sort, queue, heap
//
// Approach: Greedy algorithm with local optimal choices
// Time Complexity: O(n) to O(n^2) depending on approach
// Space Complexity: O(1) to O(n) depending on approach

func maximumNumberOfOnes(width int, height int, sideLength int, maxOnes int)
int {

}
```

Kotlin Solution:

```
class Solution {
fun maximumNumberOfOnes(width: Int, height: Int, sideLength: Int, maxOnes:
Int): Int {

}
}
```

Swift Solution:

```
class Solution {
func maximumNumberOfOnes(_ width: Int, _ height: Int, _ sideLength: Int, _
maxOnes: Int) -> Int {

}
}
```

Rust Solution:

```
// Problem: Maximum Number of Ones
// Difficulty: Hard
// Tags: greedy, math, sort, queue, heap
//
// Approach: Greedy algorithm with local optimal choices
// Time Complexity: O(n) to O(n^2) depending on approach
// Space Complexity: O(1) to O(n) depending on approach

impl Solution {
```

```
pub fn maximum_number_of_ones(width: i32, height: i32, side_length: i32,
max_ones: i32) -> i32 {

}
}
```

Ruby Solution:

```
# @param {Integer} width
# @param {Integer} height
# @param {Integer} side_length
# @param {Integer} max_ones
# @return {Integer}
def maximum_number_of_ones(width, height, side_length, max_ones)

end
```

PHP Solution:

```
class Solution {

/**
* @param Integer $width
* @param Integer $height
* @param Integer $sideLength
* @param Integer $maxOnes
* @return Integer
*/
function maximumNumberOfOnes($width, $height, $sideLength, $maxOnes) {

}
}
```

Dart Solution:

```
class Solution {
int maximumNumberOfOnes(int width, int height, int sideLength, int maxOnes) {

}
}
```

Scala Solution:

```
object Solution {
def maximumNumberOfOnes(width: Int, height: Int, sideLength: Int, maxOnes:
Int): Int = {

}
}
```

Elixir Solution:

```
defmodule Solution do
@spec maximum_number_of_ones(width :: integer, height :: integer, side_length
:: integer, max_ones :: integer) :: integer
def maximum_number_of_ones(width, height, side_length, max_ones) do

end
end
```

Erlang Solution:

```
-spec maximum_number_of_ones(Width :: integer(), Height :: integer(),
SideLength :: integer(), MaxOnes :: integer()) -> integer().
maximum_number_of_ones(Width, Height, SideLength, MaxOnes) ->
.
```

Racket Solution:

```
(define/contract (maximum-number-of-ones width height sideLength maxOnes)
(-> exact-integer? exact-integer? exact-integer? exact-integer?
exact-integer?)
)
```