Problem 835: Image Overlap

Problem Information

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

Problem Description

You are given two images,

img1

and

img2

, represented as binary, square matrices of size

n x n

. A binary matrix has only

0

s and

1

s as values.

We

translate

one image however we choose by sliding all the

1

bits left, right, up, and/or down any number of units. We then place it on top of the other image. We can then calculate the

overlap

by counting the number of positions that have a

1

in

both

images.

Note also that a translation does

not

include any kind of rotation. Any

1

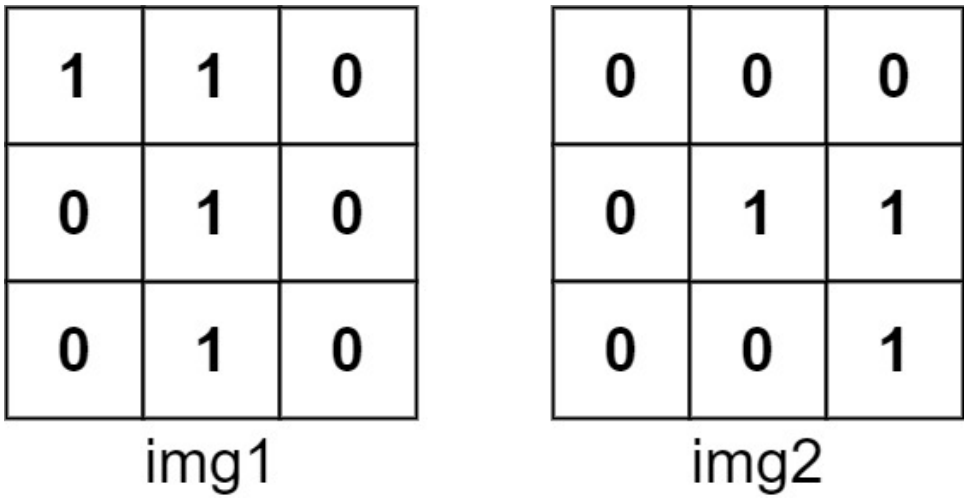
bits that are translated outside of the matrix borders are erased.

Return

the largest possible overlap

.

Example 1:



Input:

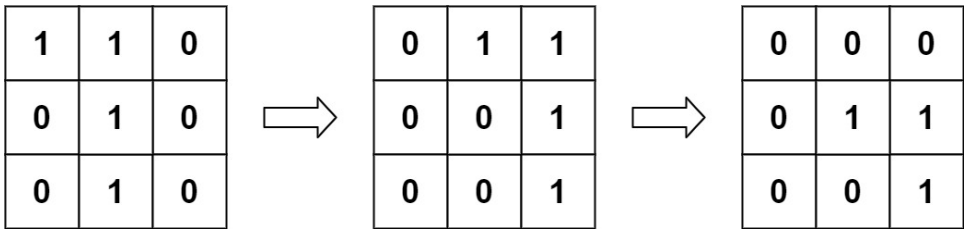
img1 = [[1,1,0],[0,1,0],[0,1,0]], img2 = [[0,0,0],[0,1,1],[0,0,1]]

Output:

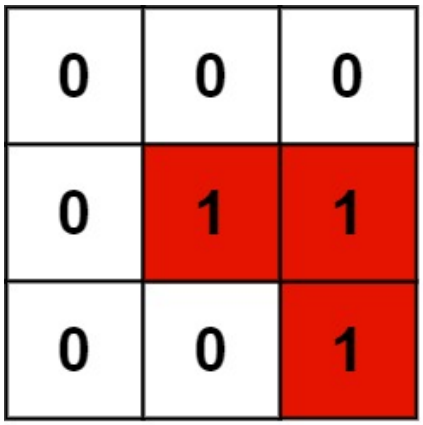
3

Explanation:

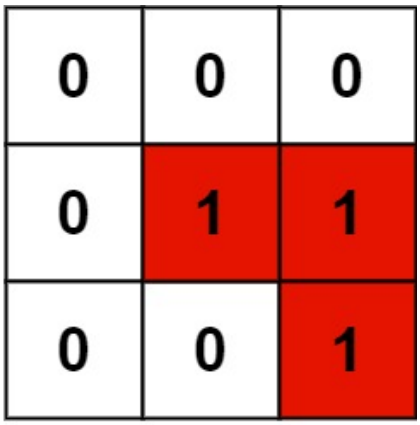
We translate img1 to right by 1 unit and down by 1 unit.



The number of positions that have a 1 in both images is 3 (shown in red).



img1



img2

Example 2:

Input:

img1 = [[1]], img2 = [[1]]

Output:

1

Example 3:

Input:

img1 = [[0]], img2 = [[0]]

Output:

0

Constraints:

n == img1.length == img1[i].length

n == img2.length == img2[i].length

1 <= n <= 30

img1[i][j]

is either

0

or

1

.

img2[i][j]

is either

0

or

1

.

Code Snippets

C++:

```
class Solution {
public:
int largestOverlap(vector<vector<int>>& img1, vector<vector<int>>& img2) {

}
};
```

Java:

```
class Solution {
public int largestOverlap(int[][] img1, int[][] img2) {
```

```
}
}
```

Python3:

```
class Solution:
def largestOverlap(self, img1: List[List[int]], img2: List[List[int]]) ->
int:
```

Python:

```
class Solution(object):
def largestOverlap(self, img1, img2):
"""
:type img1: List[List[int]]
:type img2: List[List[int]]
:rtype: int
"""
```

JavaScript:

```
/**
* @param {number[][]} img1
* @param {number[][]} img2
* @return {number}
*/
var largestOverlap = function(img1, img2) {

};
```

TypeScript:

```
function largestOverlap(img1: number[][], img2: number[][]): number {

};
```

C#:

```
public class Solution {
public int LargestOverlap(int[][] img1, int[][] img2) {
```

```
}
}
```

C:

```
int largestOverlap(int** img1, int img1Size, int* img1ColSize, int** img2,
int img2Size, int* img2ColSize) {

}
```

Go:

```
func largestOverlap(img1 [][]int, img2 [][]int) int {

}
```

Kotlin:

```
class Solution {
fun largestOverlap(img1: Array<IntArray>, img2: Array<IntArray>): Int {

}
}
```

Swift:

```
class Solution {
func largestOverlap(_ img1: [[Int]], _ img2: [[Int]]) -> Int {

}
}
```

Rust:

```
impl Solution {
pub fn largest_overlap(img1: Vec<Vec<i32>>, img2: Vec<Vec<i32>>) -> i32 {

}
}
```

Ruby:

```
# @param {Integer[][]} img1
# @param {Integer[][]} img2
# @return {Integer}
def largest_overlap(img1, img2)

end
```

PHP:

```
class Solution {

/**
* @param Integer[][] $img1
* @param Integer[][] $img2
* @return Integer
*/
function largestOverlap($img1, $img2) {

}
}
```

Dart:

```
class Solution {
int largestOverlap(List<List<int>> img1, List<List<int>> img2) {

}
}
```

Scala:

```
object Solution {
def largestOverlap(img1: Array[Array[Int]], img2: Array[Array[Int]]): Int = {

}
}
```

Elixir:

```
defmodule Solution do
@spec largest_overlap(img1 :: [[integer]], img2 :: [[integer]]) :: integer
def largest_overlap(img1, img2) do
```

```
end
end
```

Erlang:

```
-spec largest_overlap(Img1 :: [[integer()]], Img2 :: [[integer()]]) ->
integer().
largest_overlap(Img1, Img2) ->
.
```

Racket:

```
(define/contract (largest-overlap img1 img2)
(-> (listof (listof exact-integer?)) (listof (listof exact-integer?))
exact-integer?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Image Overlap
* Difficulty: Medium
* Tags: array
*
* 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 largestOverlap(vector<vector<int>>& img1, vector<vector<int>>& img2) {

}
};
```

Java Solution:

```
/**
* Problem: Image Overlap
* Difficulty: Medium
* Tags: array
*
* 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 largestOverlap(int[][] img1, int[][] img2) {

}
}
```

Python3 Solution:

```
"""
Problem: Image Overlap
Difficulty: Medium
Tags: array

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 largestOverlap(self, img1: List[List[int]], img2: List[List[int]]) ->
int:
# TODO: Implement optimized solution
pass
```

Python Solution:

```
class Solution(object):
def largestOverlap(self, img1, img2):
"""
:type img1: List[List[int]]
:type img2: List[List[int]]
:rtype: int
```

```
"""
```

JavaScript Solution:

```
/**
* Problem: Image Overlap
* Difficulty: Medium
* Tags: array
*
* 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[][]} img1
* @param {number[][]} img2
* @return {number}
*/
var largestOverlap = function(img1, img2) {

};
```

TypeScript Solution:

```
/**
* Problem: Image Overlap
* Difficulty: Medium
* Tags: array
*
* 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 largestOverlap(img1: number[][], img2: number[][]): number {

};
```

C# Solution:

```
/*
* Problem: Image Overlap
* Difficulty: Medium
* Tags: array
*
* 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 LargestOverlap(int[][] img1, int[][] img2) {

}
}
```

C Solution:

```
/*
* Problem: Image Overlap
* Difficulty: Medium
* Tags: array
*
* 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 largestOverlap(int** img1, int img1Size, int* img1ColSize, int** img2,
int img2Size, int* img2ColSize) {

}
```

Go Solution:

```
// Problem: Image Overlap
// Difficulty: Medium
// Tags: array
//
// 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 largestOverlap(img1 [][]int, img2 [][]int) int {

}
```

Kotlin Solution:

```
class Solution {
fun largestOverlap(img1: Array<IntArray>, img2: Array<IntArray>): Int {

}
}
```

Swift Solution:

```
class Solution {
func largestOverlap(_ img1: [[Int]], _ img2: [[Int]]) -> Int {

}
}
```

Rust Solution:

```
// Problem: Image Overlap
// Difficulty: Medium
// Tags: array
//
// 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 largest_overlap(img1: Vec<Vec<i32>>, img2: Vec<Vec<i32>>) -> i32 {

}
}
```

Ruby Solution:

```
# @param {Integer[][]} img1
# @param {Integer[][]} img2
```

```
# @return {Integer}
def largest_overlap(img1, img2)

end
```

PHP Solution:

```
class Solution {

/**
* @param Integer[][] $img1
* @param Integer[][] $img2
* @return Integer
*/
function largestOverlap($img1, $img2) {

}
}
```

Dart Solution:

```
class Solution {
int largestOverlap(List<List<int>> img1, List<List<int>> img2) {

}
}
```

Scala Solution:

```
object Solution {
def largestOverlap(img1: Array[Array[Int]], img2: Array[Array[Int]]): Int = {

}
}
```

Elixir Solution:

```
defmodule Solution do
@spec largest_overlap(img1 :: [[integer]], img2 :: [[integer]]) :: integer
def largest_overlap(img1, img2) do
```

```
end
end
```

Erlang Solution:

```
-spec largest_overlap(Img1 :: [[integer()]], Img2 :: [[integer()]]) ->
integer().
largest_overlap(Img1, Img2) ->
.
```

Racket Solution:

```
(define/contract (largest-overlap img1 img2)
(-> (listof (listof exact-integer?)) (listof (listof exact-integer?))
exact-integer?)
)
```