Problem 2125: Number of Laser Beams in a Bank

Problem Information

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

Problem Description

Anti-theft security devices are activated inside a bank. You are given a

0-indexed

binary string array

bank

representing the floor plan of the bank, which is an

m x n

2D matrix.

bank[i]

represents the

i

th

row, consisting of

'0'

s and

'1'

s.

'0'

means the cell is empty, while

'1'

means the cell has a security device.

There is

one

laser beam between any

two

security devices

if both

conditions are met:

The two devices are located on two

different rows

:

r

1

and

r

2

, where

r

1

< r

2

.

For

each

row

i

where

r

1

< i < r

2

, there are

no security devices

in the

i

th

row.

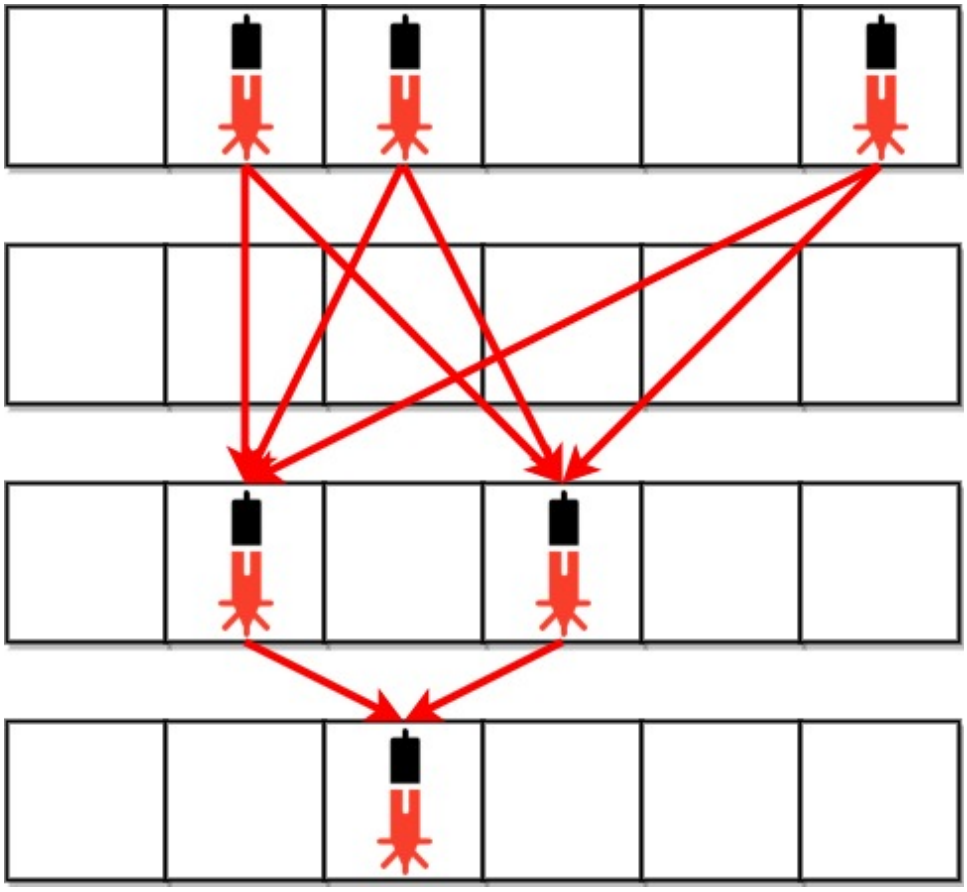
Laser beams are independent, i.e., one beam does not interfere nor join with another.

Return

the total number of laser beams in the bank

.

Example 1:



Input:

bank = ["011001","000000","010100","001000"]

Output:

8

Explanation:

Between each of the following device pairs, there is one beam. In total, there are 8 beams: * bank[0][1] -- bank[2][1] * bank[0][1] -- bank[2][3] * bank[0][2] -- bank[2][1] * bank[0][2] -- bank[2][3] * bank[0][5] -- bank[2][1] * bank[0][5] -- bank[2][3] * bank[2][1] -- bank[3][2] * bank[2][3] -- bank[3][2] Note that there is no beam between any device on the 0

th

row with any on the 3

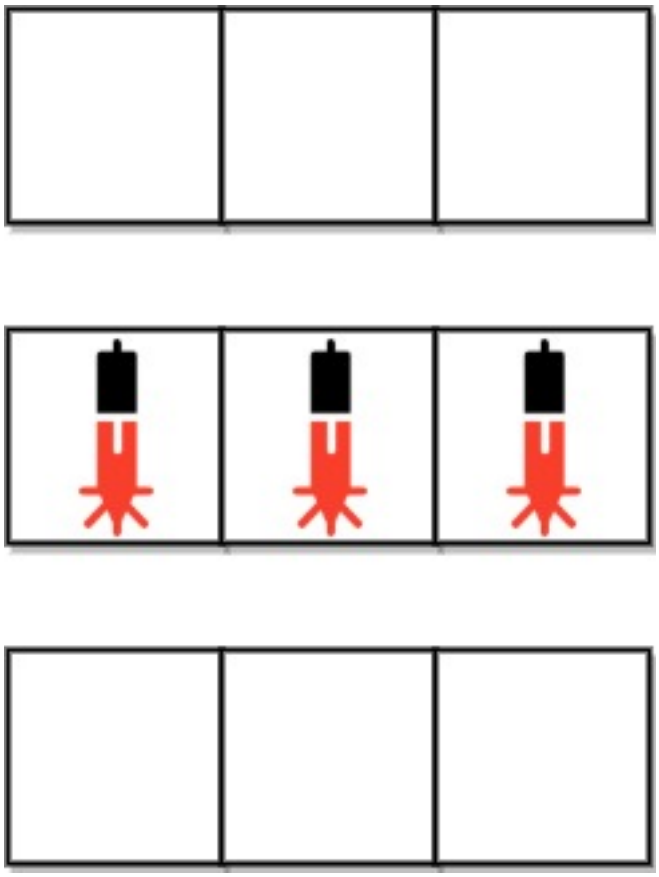
rd

row. This is because the 2

nd

row contains security devices, which breaks the second condition.

Example 2:



Input:

bank = ["000","111","000"]

Output:

0

Explanation:

There does not exist two devices located on two different rows.

Constraints:

m == bank.length

n == bank[i].length

1 <= m, n <= 500

bank[i][j]

is either

'0'

or

'1'

.

Code Snippets

C++:

```
class Solution {
public:
int numberOfBeams(vector<string>& bank) {

}
};
```

Java:

```
class Solution {
public int numberOfBeams(String[] bank) {

}
}
```

Python3:

```
class Solution:
def numberOfBeams(self, bank: List[str]) -> int:
```

Python:

```
class Solution(object):
def numberOfBeams(self, bank):
```

```
"""
:type bank: List[str]
:rtype: int
"""
```

JavaScript:

```
/**
* @param {string[]} bank
* @return {number}
*/
var numberOfBeams = function(bank) {

};
```

TypeScript:

```
function numberOfBeams(bank: string[]): number {

};
```

C#:

```
public class Solution {
public int NumberOfBeams(string[] bank) {

}
}
```

C:

```
int numberOfBeams(char** bank, int bankSize) {

}
```

Go:

```
func numberOfBeams(bank []string) int {

}
```

Kotlin:

```
class Solution {
fun numberOfBeams(bank: Array<String>): Int {

}
}
```

Swift:

```
class Solution {
func numberOfBeams(_ bank: [String]) -> Int {

}
}
```

Rust:

```
impl Solution {
pub fn number_of_beams(bank: Vec<String>) -> i32 {

}
}
```

Ruby:

```
# @param {String[]} bank
# @return {Integer}
def number_of_beams(bank)

end
```

PHP:

```
class Solution {

/**
* @param String[] $bank
* @return Integer
*/
function numberOfBeams($bank) {

}
}
```

Dart:

```
class Solution {
int numberOfBeams(List<String> bank) {

}
}
```

Scala:

```
object Solution {
def numberOfBeams(bank: Array[String]): Int = {

}
}
```

Elixir:

```
defmodule Solution do
@spec number_of_beams(bank :: [String.t]) :: integer
def number_of_beams(bank) do

end
end
```

Erlang:

```
-spec number_of_beams(Bank :: [unicode:unicode_binary()]) -> integer().
number_of_beams(Bank) ->
.
```

Racket:

```
(define/contract (number-of-beams bank)
(-> (listof string?) exact-integer?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Number of Laser Beams in a Bank
* Difficulty: Medium
* Tags: array, string, math
*
* 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 numberOfBeams(vector<string>& bank) {

}
};
```

Java Solution:

```
/**
* Problem: Number of Laser Beams in a Bank
* Difficulty: Medium
* Tags: array, string, math
*
* 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 numberOfBeams(String[] bank) {

}
}
```

Python3 Solution:

```
"""
Problem: Number of Laser Beams in a Bank
Difficulty: Medium
Tags: array, string, math
```

```
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 numberOfBeams(self, bank: List[str]) -> int:
# TODO: Implement optimized solution
pass
```

Python Solution:

```
class Solution(object):
def numberOfBeams(self, bank):
"""
:type bank: List[str]
:rtype: int
"""
```

JavaScript Solution:

```
/**
* Problem: Number of Laser Beams in a Bank
* Difficulty: Medium
* Tags: array, string, math
*
* 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 {string[]} bank
* @return {number}
*/
var numberOfBeams = function(bank) {

};
```

TypeScript Solution:

```
/**
* Problem: Number of Laser Beams in a Bank
* Difficulty: Medium
* Tags: array, string, math
*
* 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 numberOfBeams(bank: string[]): number {

};
```

C# Solution:

```
/*
* Problem: Number of Laser Beams in a Bank
* Difficulty: Medium
* Tags: array, string, math
*
* 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 NumberOfBeams(string[] bank) {

}
}
```

C Solution:

```
/*
* Problem: Number of Laser Beams in a Bank
* Difficulty: Medium
* Tags: array, string, math
*
* 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 numberOfBeams(char** bank, int bankSize) {

}
```

Go Solution:

```
// Problem: Number of Laser Beams in a Bank
// Difficulty: Medium
// Tags: array, string, math
//
// 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 numberOfBeams(bank []string) int {

}
```

Kotlin Solution:

```
class Solution {
fun numberOfBeams(bank: Array<String>): Int {

}
}
```

Swift Solution:

```
class Solution {
func numberOfBeams(_ bank: [String]) -> Int {

}
}
```

Rust Solution:

```
// Problem: Number of Laser Beams in a Bank
// Difficulty: Medium
// Tags: array, string, math
```

```
//
// 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 number_of_beams(bank: Vec<String>) -> i32 {

}
}
```

Ruby Solution:

```
# @param {String[]} bank
# @return {Integer}
def number_of_beams(bank)

end
```

PHP Solution:

```
class Solution {

/**
* @param String[] $bank
* @return Integer
*/
function numberOfBeams($bank) {

}
}
```

Dart Solution:

```
class Solution {
int numberOfBeams(List<String> bank) {

}
}
```

Scala Solution:

```
object Solution {
def numberOfBeams(bank: Array[String]): Int = {

}
}
```

Elixir Solution:

```
defmodule Solution do
@spec number_of_beams(bank :: [String.t]) :: integer
def number_of_beams(bank) do

end
end
```

Erlang Solution:

```
-spec number_of_beams(Bank :: [unicode:unicode_binary()]) -> integer().
number_of_beams(Bank) ->
.
```

Racket Solution:

```
(define/contract (number-of-beams bank)
(-> (listof string?) exact-integer?)
)
```