Problem 934: Shortest Bridge

Problem Information

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

Problem Description

You are given an

n x n

binary matrix

grid

where

1

represents land and

0

represents water.

An

island

is a 4-directionally connected group of

1

's not connected to any other

1

's. There are

exactly two islands

in

grid

.

You may change

0

's to

1

's to connect the two islands to form

one island

.

Return

the smallest number of

0

's you must flip to connect the two islands

.

Example 1:

Input:

grid = [[0,1],[1,0]]

Output:

1

Example 2:

Input:

grid = [[0,1,0],[0,0,0],[0,0,1]]

Output:

2

Example 3:

Input:

grid = [[1,1,1,1,1],[1,0,0,0,1],[1,0,1,0,1],[1,0,0,0,1],[1,1,1,1,1]]

Output:

1

Constraints:

n == grid.length == grid[i].length

2 <= n <= 100

grid[i][j]

is either

0

or

1

.

There are exactly two islands in

grid

.

Code Snippets

C++:

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

}
};
```

Java:

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

}
}
```

Python3:

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

Python:

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

JavaScript:

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

};
```

TypeScript:

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

};
```

C#:

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

}
}
```

C:

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

}
```

Go:

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

}
```

Kotlin:

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

}
}
```

Swift:

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

}
}
```

Rust:

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

}
}
```

Ruby:

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

end
```

PHP:

```
class Solution {

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

}
```

```
}
```

Dart:

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

}
}
```

Scala:

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

}
}
```

Elixir:

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

end
end
```

Erlang:

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

Racket:

```
(define/contract (shortest-bridge grid)
(-> (listof (listof exact-integer?)) exact-integer?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Shortest Bridge
* Difficulty: Medium
* Tags: array, search
*
* 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 shortestBridge(vector<vector<int>>& grid) {

}
};
```

Java Solution:

```
/**
* Problem: Shortest Bridge
* Difficulty: Medium
* Tags: array, search
*
* 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 shortestBridge(int[][] grid) {

}
}
```

Python3 Solution:

```
"""
Problem: Shortest Bridge
Difficulty: Medium
Tags: array, search
```

```
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 shortestBridge(self, grid: List[List[int]]) -> int:
# TODO: Implement optimized solution
pass
```

Python Solution:

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

JavaScript Solution:

```
/**
* Problem: Shortest Bridge
* Difficulty: Medium
* Tags: array, search
*
* 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 shortestBridge = function(grid) {

};
```

TypeScript Solution:

```
/**
* Problem: Shortest Bridge
* Difficulty: Medium
* Tags: array, search
*
* 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 shortestBridge(grid: number[][]): number {

};
```

C# Solution:

```
/*
* Problem: Shortest Bridge
* Difficulty: Medium
* Tags: array, search
*
* 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 ShortestBridge(int[][] grid) {

}
}
```

C Solution:

```
/*
* Problem: Shortest Bridge
* Difficulty: Medium
* Tags: array, search
*
* 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 shortestBridge(int** grid, int gridSize, int* gridColSize) {

}
```

Go Solution:

```
// Problem: Shortest Bridge
// Difficulty: Medium
// Tags: array, search
//
// 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 shortestBridge(grid [][]int) int {

}
```

Kotlin Solution:

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

}
}
```

Swift Solution:

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

}
}
```

Rust Solution:

```
// Problem: Shortest Bridge
// Difficulty: Medium
// Tags: array, search
```

```
//
// 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 shortest_bridge(grid: Vec<Vec<i32>>) -> i32 {

}
}
```

Ruby Solution:

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

end
```

PHP Solution:

```
class Solution {

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

}
}
```

Dart Solution:

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

}
}
```

Scala Solution:

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

}
}
```

Elixir Solution:

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

end
end
```

Erlang Solution:

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

Racket Solution:

```
(define/contract (shortest-bridge grid)
(-> (listof (listof exact-integer?)) exact-integer?)
)
```