Problem 3071: Minimum Operations to Write the Letter Y on a Grid

Problem Information

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

Problem Description

You are given a

0-indexed

n x n

grid where

n

is odd, and

grid[r][c]

is

0

,

1

, or

2

.

We say that a cell belongs to the Letter

Y

if it belongs to one of the following:

The diagonal starting at the top-left cell and ending at the center cell of the grid.

The diagonal starting at the top-right cell and ending at the center cell of the grid.

The vertical line starting at the center cell and ending at the bottom border of the grid.

The Letter

Y

is written on the grid if and only if:

All values at cells belonging to the Y are equal.

All values at cells not belonging to the Y are equal.

The values at cells belonging to the Y are different from the values at cells not belonging to the Y.

Return

the

minimum

number of operations needed to write the letter Y on the grid given that in one operation you can change the value at any cell to

0

,

1

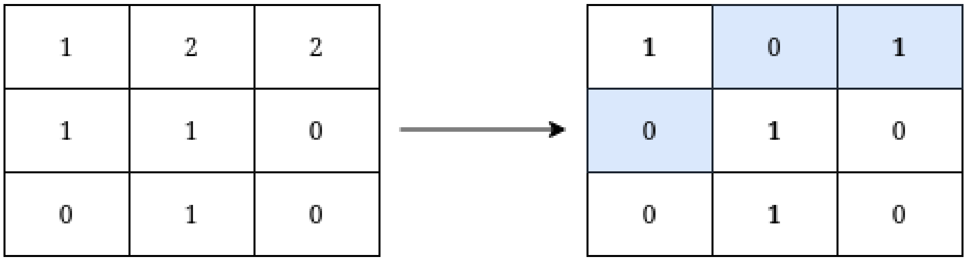
,

or

2

.

Example 1:



1	2	2
1	1	0
0	1	0

1	0	**1**
0	**1**	0
0	**1**	0

Input:

grid = [[1,2,2],[1,1,0],[0,1,0]]

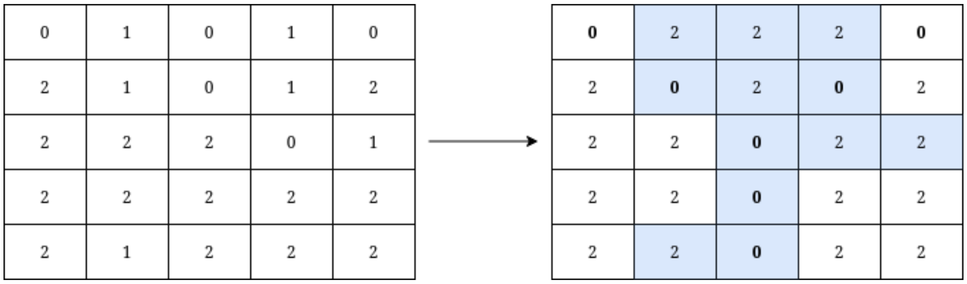
Output:

3

Explanation:

We can write Y on the grid by applying the changes highlighted in blue in the image above. After the operations, all cells that belong to Y, denoted in bold, have the same value of 1 while those that do not belong to Y are equal to 0. It can be shown that 3 is the minimum number of operations needed to write Y on the grid.

Example 2:



0	1	0	1	0
2	1	0	1	2
2	2	2	0	1
2	2	2	2	2
2	1	2	2	2

0	2	2	2	**0**
2	**0**	2	**0**	2
2	2	**0**	2	2
2	2	**0**	2	2
2	2	**0**	2	2

Input:

grid = [[0,1,0,1,0],[2,1,0,1,2],[2,2,2,0,1],[2,2,2,2,2],[2,1,2,2,2]]

Output:

12

Explanation:

We can write Y on the grid by applying the changes highlighted in blue in the image above. After the operations, all cells that belong to Y, denoted in bold, have the same value of 0 while those that do not belong to Y are equal to 2. It can be shown that 12 is the minimum number of operations needed to write Y on the grid.

Constraints:

3 <= n <= 49

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

0 <= grid[i][j] <= 2

n

is odd.

Code Snippets

C++:

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

}
};
```

Java:

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

}
}
```

Python3:

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

Python:

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

JavaScript:

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

};
```

TypeScript:

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

};
```

C#:

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

}
}
```

C:

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

}
```

Go:

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

}
```

Kotlin:

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

}
}
```

Swift:

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

}
}
```

Rust:

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

Ruby:

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

end
```

PHP:

```
class Solution {

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

}
}
```

Dart:

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

}
}
```

Scala:

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

}
}
```

Elixir:

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

end
end
```

Erlang:

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

Racket:

```
(define/contract (minimum-operations-to-write-y grid)
(-> (listof (listof exact-integer?)) exact-integer?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Minimum Operations to Write the Letter Y on a Grid
* Difficulty: Medium
* Tags: array, hash
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/

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

}
};
```

Java Solution:

```
/**
* Problem: Minimum Operations to Write the Letter Y on a Grid
* Difficulty: Medium
* Tags: array, hash
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/

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

}
}
```

Python3 Solution:

```
"""
Problem: Minimum Operations to Write the Letter Y on a Grid
Difficulty: Medium
Tags: array, hash

Approach: Use two pointers or sliding window technique
Time Complexity: O(n) or O(n log n)
Space Complexity: O(n) for hash map
"""

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

Python Solution:

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

JavaScript Solution:

```
/**
* Problem: Minimum Operations to Write the Letter Y on a Grid
* Difficulty: Medium
* Tags: array, hash
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/

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

};
```

TypeScript Solution:

```
/**
* Problem: Minimum Operations to Write the Letter Y on a Grid
* Difficulty: Medium
* Tags: array, hash
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/

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

};
```

C# Solution:

```
/*
* Problem: Minimum Operations to Write the Letter Y on a Grid
* Difficulty: Medium
* Tags: array, hash
*
```

```
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/

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

}
}
```

C Solution:

```
/*
* Problem: Minimum Operations to Write the Letter Y on a Grid
* Difficulty: Medium
* Tags: array, hash
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/

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

}
```

Go Solution:

```
// Problem: Minimum Operations to Write the Letter Y on a Grid
// Difficulty: Medium
// Tags: array, hash
//
// Approach: Use two pointers or sliding window technique
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(n) for hash map

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

}
```

Kotlin Solution:

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

}
}
```

Swift Solution:

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

}
}
```

Rust Solution:

```
// Problem: Minimum Operations to Write the Letter Y on a Grid
// Difficulty: Medium
// Tags: array, hash
//
// Approach: Use two pointers or sliding window technique
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(n) for hash map

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

}
}
```

Ruby Solution:

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

end
```

PHP Solution:

```
class Solution {

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

}
}
```

Dart Solution:

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

}
}
```

Scala Solution:

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

}
}
```

Elixir Solution:

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

end
end
```

Erlang Solution:

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

Racket Solution:

```
(define/contract (minimum-operations-to-write-y grid)
(-> (listof (listof exact-integer?)) exact-integer?)
)
```