Problem 2510: Check if There is a Path With Equal Number of 0's And 1's

Problem Information

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

Problem Description

You are given a

0-indexed

m x n

binary

matrix

grid

. You can move from a cell

(row, col)

to any of the cells

(row + 1, col)

or

(row, col + 1)

.

Return

true

if there is a path from

(0, 0)

to

(m - 1, n - 1)

that visits an

equal

number of

0

's and

1

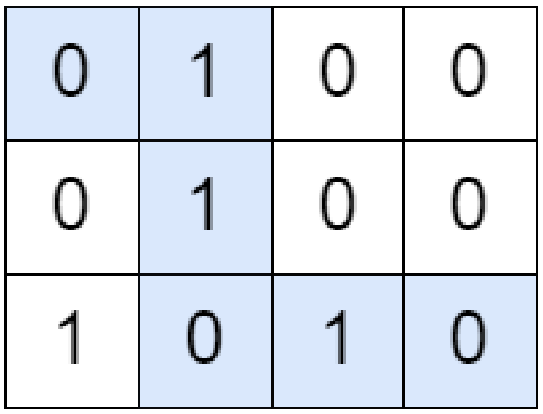
's

. Otherwise return

false

.

Example 1:



Input:

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

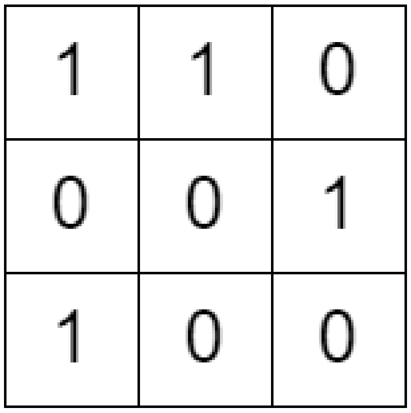
Output:

true

Explanation:

The path colored in blue in the above diagram is a valid path because we have 3 cells with a value of 1 and 3 with a value of 0. Since there is a valid path, we return true.

Example 2:



Input:

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

Output:

false

Explanation:

There is no path in this grid with an equal number of 0's and 1's.

Constraints:

m == grid.length

n == grid[i].length

2 <= m, n <= 100

grid[i][j]

is either

0

or

1

.

Code Snippets

C++:

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

}
};
```

Java:

```
class Solution {
public boolean isThereAPath(int[][] grid) {

}
}
```

Python3:

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

Python:

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

JavaScript:

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

};
```

TypeScript:

```
function isThereAPath(grid: number[][]): boolean {

};
```

C#:

```
public class Solution {
public bool IsThereAPath(int[][] grid) {

}
}
```

C:

```
bool isThereAPath(int** grid, int gridSize, int* gridColSize) {

}
```

Go:

```
func isThereAPath(grid [][]int) bool {

}
```

Kotlin:

```
class Solution {
fun isThereAPath(grid: Array<IntArray>): Boolean {

}
}
```

Swift:

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

}
}
```

Rust:

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

}
}
```

Ruby:

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

end
```

PHP:

```
class Solution {

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

}
}
```

Dart:

```
class Solution {
bool isThereAPath(List<List<int>> grid) {

}
}
```

Scala:

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

}
}
```

Elixir:

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

end
end
```

Erlang:

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

Racket:

```
(define/contract (is-there-a-path grid)
(-> (listof (listof exact-integer?)) boolean?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Check if There is a Path With Equal Number of 0's And 1's
* Difficulty: Medium
* Tags: array, dp
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) or O(n * m) for DP table
*/

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

}
};
```

Java Solution:

```
/**
* Problem: Check if There is a Path With Equal Number of 0's And 1's
* Difficulty: Medium
* Tags: array, dp
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) or O(n * m) for DP table
*/

class Solution {
public boolean isThereAPath(int[][] grid) {
}
}
```

Python3 Solution:

```
"""
Problem: Check if There is a Path With Equal Number of 0's And 1's
Difficulty: Medium
Tags: array, dp

Approach: Use two pointers or sliding window technique
Time Complexity: O(n) or O(n log n)
Space Complexity: O(n) or O(n * m) for DP table
"""

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

Python Solution:

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

JavaScript Solution:

```
/**
* Problem: Check if There is a Path With Equal Number of 0's And 1's
* Difficulty: Medium
* Tags: array, dp
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) or O(n * m) for DP table
*/
```

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

};
```

TypeScript Solution:

```
/**
* Problem: Check if There is a Path With Equal Number of 0's And 1's
* Difficulty: Medium
* Tags: array, dp
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) or O(n * m) for DP table
*/

function isThereAPath(grid: number[][]): boolean {

};
```

C# Solution:

```
/*
* Problem: Check if There is a Path With Equal Number of 0's And 1's
* Difficulty: Medium
* Tags: array, dp
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) or O(n * m) for DP table
*/

public class Solution {
public bool IsThereAPath(int[][] grid) {

}
}
```

C Solution:

```
/*
* Problem: Check if There is a Path With Equal Number of 0's And 1's
* Difficulty: Medium
* Tags: array, dp
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) or O(n * m) for DP table
*/

bool isThereAPath(int** grid, int gridSize, int* gridColSize) {

}
```

Go Solution:

```
// Problem: Check if There is a Path With Equal Number of 0's And 1's
// Difficulty: Medium
// Tags: array, dp
//
// Approach: Use two pointers or sliding window technique
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(n) or O(n * m) for DP table

func isThereAPath(grid [][]int) bool {

}
```

Kotlin Solution:

```
class Solution {
fun isThereAPath(grid: Array<IntArray>): Boolean {

}
}
```

Swift Solution:

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

}
}
```

Rust Solution:

```
// Problem: Check if There is a Path With Equal Number of 0's And 1's
// Difficulty: Medium
// Tags: array, dp
//
// Approach: Use two pointers or sliding window technique
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(n) or O(n * m) for DP table

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

}
}
```

Ruby Solution:

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

end
```

PHP Solution:

```
class Solution {

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

}
}
```

Dart Solution:

```
class Solution {
bool isThereAPath(List<List<int>> grid) {

}
}
```

Scala Solution:

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

}
}
```

Elixir Solution:

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

end
end
```

Erlang Solution:

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

Racket Solution:

```
(define/contract (is-there-a-path grid)
(-> (listof (listof exact-integer?)) boolean?)
)
```