1905. Count Sub Islands

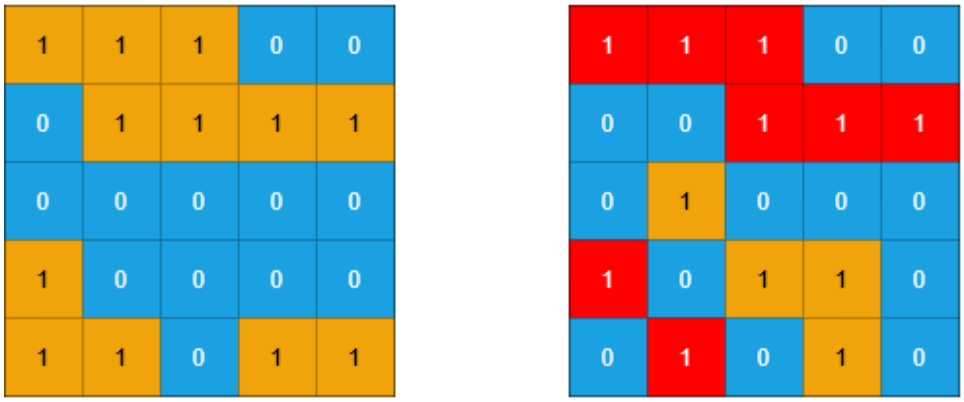
Created	@November 29, 2022 4:50 PM
Difficulty	Medium
LC Url	https://leetcode.com/problems/count-sub-islands/
Importance	
Tag	DFS Island
Reference	https://labuladong.github.io/algo/4/31/107/

You are given two `m x n` binary matrices `grid1` and `grid2` containing only `0`'s (representing water) and `1`'s (representing land). An **island** is a group of `1`'s connected **4-directionally** (horizontal or vertical). Any cells outside of the grid are considered water cells.

An island in `grid2` is considered a **sub-island** if there is an island in `grid1` that contains **all** the cells that make up **this** island in `grid2`.

Return the ***number*** *of islands in* `grid2` *that are considered* ***sub-islands***.

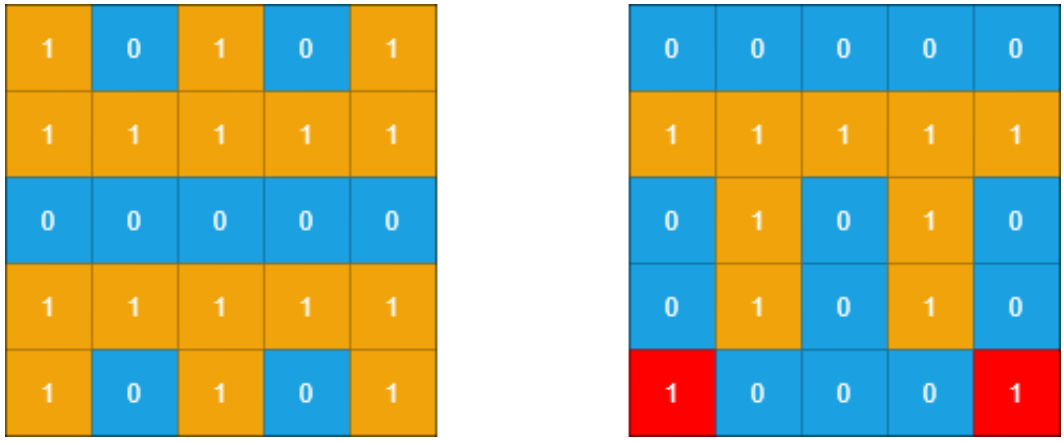
Example 1:




```
Input: grid1 = [[1,1,1,0,0],[0,1,1,1,1],[0,0,0,0,0],[1,0,0,0,0],[1,1,0,1,1]], grid2 = [[1,
1,1,0,0],[0,0,1,1,1],[0,1,0,0,0],[1,0,1,1,0],[0,1,0,1,0]]
Output: 3
Explanation:In the picture above, the grid on the left is grid1 and the grid on the right
```

```
 is grid2.
The 1s colored red in grid2 are those considered to be part of a sub-island. There are thr
ee sub-islands.
```

Example 2:




```
Input: grid1 = [[1,0,1,0,1],[1,1,1,1,1],[0,0,0,0,0],[1,1,1,1,1],[1,0,1,0,1]], grid2 = [[0,
0,0,0,0],[1,1,1,1,1],[0,1,0,1,0],[0,1,0,1,0],[1,0,0,0,1]]
Output: 2
Explanation:In the picture above, the grid on the left is grid1 and the grid on the right
 is grid2.
The 1s colored red in grid2 are those considered to be part of a sub-island. There are two
sub-islands.
```

Constraints:

- `m == grid1.length == grid2.length`
- `n == grid1[i].length == grid2[i].length`
- `1 <= m, n <= 500`
- `grid1[i][j]` and `grid2[i][j]` are either `0` or `1`.

Solution

```
class Solution:
    directions = [(0, 1), (0, -1), (1, 0), (-1, 0)]

    def countSubIslands(self, grid1: List[List[int]], grid2: List[List[int]]) -> int:
```

```
        res = 0
        m = len(grid1)
        if m == 0:
            return res
        n = len(grid1[0])

        for i in range(m):
            for j in range(n):
                if grid1[i][j] == 0 and grid2[i][j] == 1:
                    # this island must not be a sub island
                    # so we change it to sea
                    self.dfs(grid2, i, j)

        # now, every islands is sub island
        # count the number of islands for grid2
        for i in range(m):
            for j in range(n):
                if grid2[i][j] == 1:
                    res += 1
                    self.dfs(grid2, i, j)

        return res

    def dfs(self, grid, i, j):
        if not self.is_valid(grid, i, j):
            return

        if grid[i][j] == 0:
            return

        grid[i][j] = 0

        for direction in self.directions:
            cur_i, cur_j = i + direction[0], j +direction[1]
            self.dfs(grid, cur_i, cur_j)

    def is_valid(self, grid, i, j):
        m, n = len(grid), len(grid[0])
        if 0 <= i < m and 0 <= j < n:
            return True
        return False
```