Algorithms

DFS Homework 2

Mostafa S. Ibrahim
Teaching, Training and Coaching for more than a decade!

Artificial Intelligence & Computer Vision Researcher
PhD from Simon Fraser University - Canada
Bachelor / Msc from Cairo University - Egypt
Ex-(Software Engineer / ICPC World Finalist)

Problem #1: LeetCode 1905 - Count Sub Islands

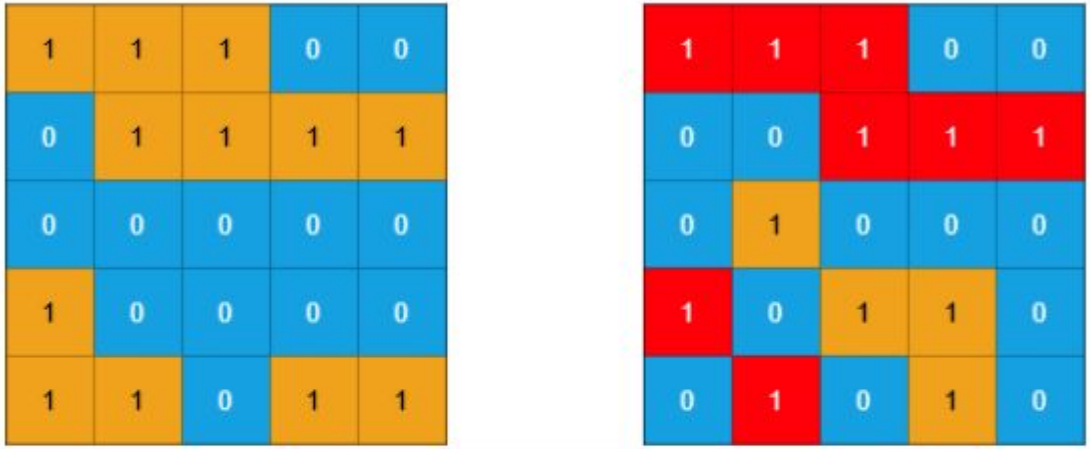
You are given two `m x n` binary matrices `grid1` and `grid2` containing only `0` 's (representing water) and `1` 's (representing land). An **island** is a group of `1` 's connected **4-directionally** (horizontal or vertical). Any cells outside of the grid are considered water cells.

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***.

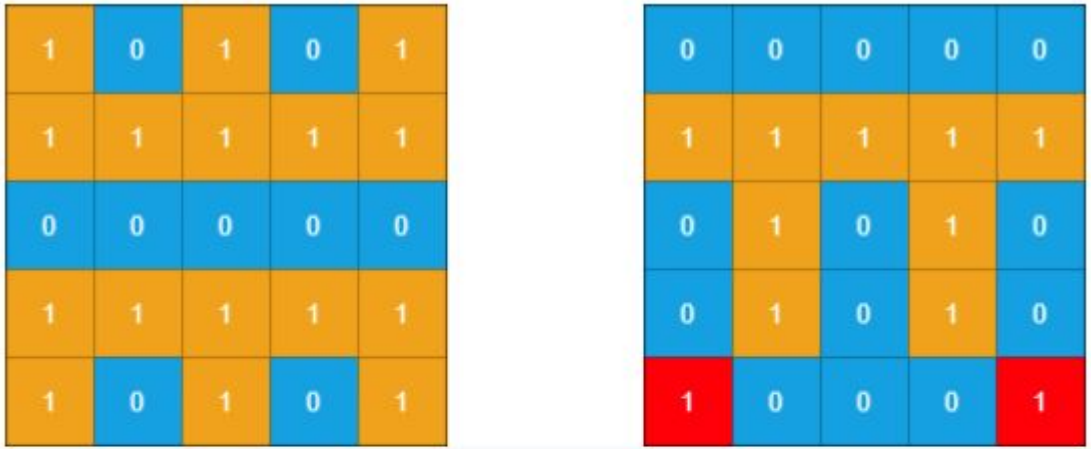
- int countSubIslands(vector<vector<int>> &grid1, vector<vector<int>> &grid2)
  - m == grid1.length == grid2.length
  - n == grid1[i].length == grid2[i].length
  - 1 <= m, n <= 500
  - grid1[i][j] and grid2[i][j] are either 0 or 1.

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 three 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.

Problem #2: LeetCode 1034 - Coloring A Border

You are given an `m x n` integer matrix `grid`, and three integers `row`, `col`, and `color`. Each value in the grid represents the color of the grid square at that location.

Two squares belong to the same **connected component** if they have the same color and are next to each other in any of the 4 directions.

The **border of a connected component** is all the squares in the connected component that are either **4-directionally** adjacent to a square not in the component, or on the boundary of the grid (the first or last row or column).

You should color the **border** of the **connected component** that contains the square `grid[row][col]` with `color`.

Return *the final grid*.

- vector<vector<int>> colorBorder(vector<vector<int>> &grid, int sr, int sc, int newColor)
  - 1 <= grid[i][j], color <= 1000

Example 1:

```
Input: grid = [[1,1],[1,2]], row = 0, col = 0, color = 3
Output: [[3,3],[3,2]]
```

Example 2:

```
Input: grid = [[1,2,2],[2,3,2]], row = 0, col = 1, color = 3
Output: [[1,3,3],[2,3,3]]
```

Example 3:

```
Input: grid = [[1,1,1],[1,1,1],[1,1,1]], row = 1, col = 1, color = 2
Output: [[2,2,2],[2,1,2],[2,2,2]]
```

My example

- Assume start (0, 0)
- The colored cells should be replaced
- Just apply the rules
  - adjacent to a square not in the component,
  - or on the boundary of the grid

2	3	3	2	3
2	2	2	2	3
2	2	2	2	3
2	2	4	2	3
2	2	2	2	3

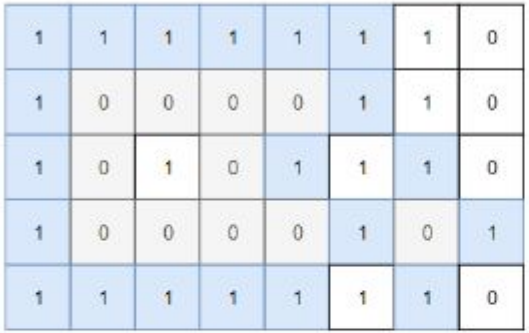
Problem #3: LeetCode 1254 - Number of Closed Islands

Given a 2D `grid` consists of `0s` (land) and `1s` (water). An *island* is a maximal 4-directionally connected group of `0s` and a *closed island* is an island **totally** (all left, top, right, bottom) surrounded by `1s.`

Return the number of *closed islands*.

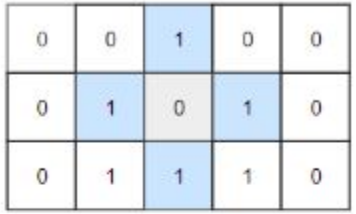
- int closedIsland(vector<vector<int>> &grid)

Example 1:



Input: grid = [[1,1,1,1,1,1,1,0],[1,0,0,0,0,1,1,0],
[1,0,1,0,1,1,1,0],[1,0,0,0,0,1,0,1],[1,1,1,1,1,1,1,0]]
Output: 2
Explanation:
Islands in gray are closed because they are completely surrounded by water (group of 1s).

Example 2:




```
Input: grid = [[0,0,1,0,0],[0,1,0,1,0],[0,1,1,1,0]]
Output: 1
```

Example 3:

```
Input: grid = [[1,1,1,1,1,1,1],
               [1,0,0,0,0,0,1],
               [1,0,1,1,1,0,1],
               [1,0,1,0,1,0,1],
               [1,0,1,1,1,0,1],
               [1,0,0,0,0,0,1],
               [1,1,1,1,1,1,1]]
Output: 2
```

Problem #4: LeetCode 1559 - Detect Cycles in 2D Grid

Given a 2D array of characters `grid` of size `m x n`, you need to find if there exists any cycle consisting of the **same value** in `grid`.

A cycle is a path of **length 4 or more** in the grid that starts and ends at the same cell. From a given cell, you can move to one of the cells adjacent to it - in one of the four directions (up, down, left, or right), if it has the **same value** of the current cell.

Also, you cannot move to the cell that you visited in your last move. For example, the cycle `(1, 1) -> (1, 2) -> (1, 1)` is invalid because from `(1, 2)` we visited `(1, 1)` which was the last visited cell.

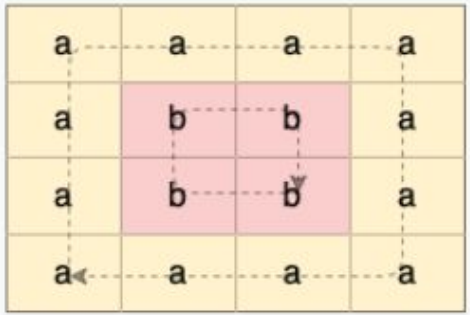
Return `true` if any cycle of the same value exists in `grid`, otherwise, return `false`.

- bool containsCycle(vector<vector<char>> &grid)

Example 1:

a	a	a	a
a	b	b	a
a	b	b	a
a	a	a	a

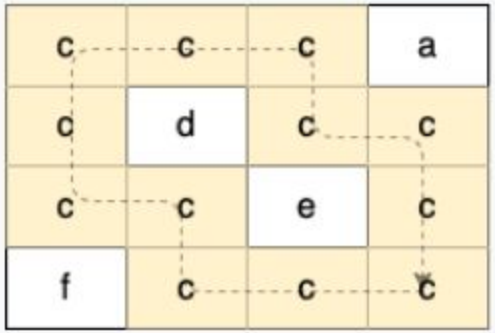
Input: grid = [["a","a","a","a"],["a","b","b","a"],["a","b","b","a"],["a","a","a","a"]]
Output: true
Explanation: There are two valid cycles shown in different colors in the image below:



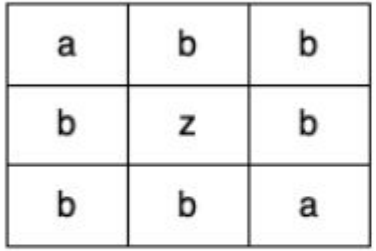
Example 2:

c	c	c	a
c	d	c	c
c	c	e	c
f	c	c	c

Input: grid = [["c","c","c","a"],["c","d","c","c"],["c","c","e","c"],["f","c","c","c"]]
Output: true
Explanation: There is only one valid cycle highlighted in the image below:



Example 3:




```
Input: grid = [["a","b","b"],["b","z","b"],["b","b","a"]]
Output: false
```

"Acquire knowledge and impart it to the people."

"Seek knowledge from the Cradle to the Grave."