733. Flood Fill

An image is represented by an m x n integer
grid image where image[i][j] represents the pixel value of the image.

You are also given three integers sr, sc, and color. You should perform a **flood fill** on the image starting from the pixel image[sr][sc].

To perform a **flood fill**, consider the starting pixel, plus any pixels connected **4-directionally** to the starting pixel of the same color as the starting pixel, plus any pixels connected **4-directionally** to those pixels (also with the same color), and so on. Replace the color of all of the aforementioned pixels with color.

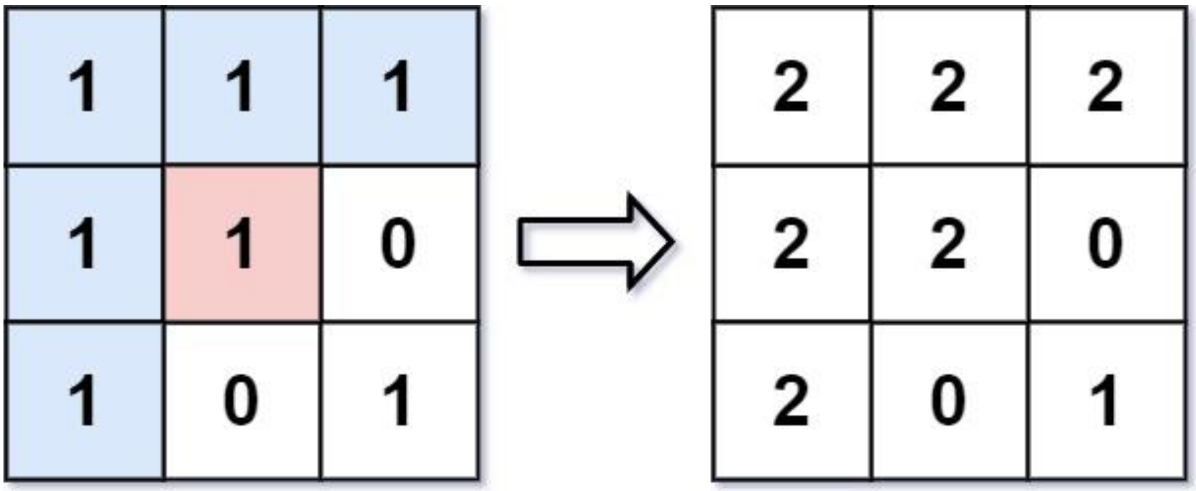
Return *the modified image after performing the flood fill*.

Example 1:

Input: image = [[1,1,1],[1,1,0],[1,0,1]], sr = 1, sc = 1, color = 2

Output: [[2,2,2],[2,2,0],[2,0,1]]

Explanation:



From the center of the image with position (sr, sc) = (1, 1) (i.e., the red pixel), all pixels connected by a path of the same color as the starting pixel (i.e., the blue pixels) are colored with the new color.

Note the bottom corner is **not** colored 2, because it is not 4-directionally connected to the starting pixel.

Example 2:

Input: image = [[0,0,0],[0,0,0]], sr = 0, sc = 0, color = 0

Output: [[0,0,0],[0,0,0]]

Explanation:

The starting pixel is already colored with 0, which is the same as the target color. Therefore, no changes are made to the image.

Constraints:

- m == image.length
- n == image[i].length
- 1 <= m, n <= 50
- 0 <= image[i][j], color < 2^{16}
- 0 <= sr < m
- 0 <= sc < n

```
class Solution(object):
    def floodFill(self, image, sr, sc, color):
        m, n = len(image), len(image[0])
        original_color = image[sr][sc]
        if original_color == color:
            return image

        def dfs(x, y):
            if x < 0 or y < 0 or x >= m or y >= n or
image[x][y] != original_color:
                return
            image[x][y] = color
            dfs(x+1, y)
            dfs(x-1, y)
            dfs(x, y+1)
            dfs(x, y-1)

        dfs(sr, sc)
        return image
```