1034. Coloring A Border

ou 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*.

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]]
```

Constraints:

- `m == grid.length`
- `n == grid[i].length`
- `1 <= m, n <= 50`
- `1 <= grid[i][j], color <= 1000`
- `0 <= row < m`
- `0 <= col < n`

```python
class Solution:
    def colorBorder(self, grid: List[List[int]], row: int, col: int, color: int) -> List[List[int]]:

        self.dfs(grid,row,col,grid[row][col])
        for i in range(len(grid)):
            for j in range(len(grid[0])):
                if grid[i][j]<0:
                    grid[i][j]=color
        return grid




    def dfs(self,grid,x,y,color):
        grid[x][y]=-color
        count = 0
        for dx,dy in ([0,1],[1,0],[0,-1],[-1,0]):
            xx = x+dx
            yy = y+dy

            if xx>=len(grid) or yy>=len(grid[0]) or xx<0 or yy<0 or abs(grid[xx][yy])!=color:
                continue
            count=count+1
            if grid[xx][yy]==color:
                self.dfs(grid,xx,yy,color)

        if count==4:
            grid[x][y]=color
```