Flood Fill

Solution

An `image` is represented by a 2-D array of integers, each integer representing the pixel value of the image (from 0 to 65535).

Given a coordinate `(sr, sc)` representing the starting pixel (row and column) of the flood fill, and a pixel value `newColor`, "flood fill" the image.

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 as the starting pixel), and so on. Replace the color of all of the aforementioned pixels with the newColor.

At the end, return the modified image.

Example 1:

```
Input:
image = [[1,1,1],[1,1,0],[1,0,1]]
sr = 1, sc = 1, newColor = 2
Output: [[2,2,2],[2,2,0],[2,0,1]]
Explanation:
Note the bottom corner is not colored 2, because it is not 4-directionally connected
to the starting pixel.
```

Note:

- The length of `image` and `image[0]` will be in the range `[1, 50]`.
- The given starting pixel will satisfy `0 <= sr < image.length` and `0 <= sc < image[0].length`.
- The value of each color in `image[i][j]` and `newColor` will be an integer in `[0, 65535]`.

Show Hint #1

Java

```java
class Solution {
    public int[][] floodFill(int[][] image, int sr, int sc, int newColor) {
    public int[][] floodFill(int[][] image, int sr, int sc, int newColor) {
        if(image.length==0 || image[sr][sc]==newColor) return image;
        dfs(image,sr,sc,newColor,image[sr][sc]);
        return image;
    }
    public void dfs(int[][] image,int r,int c,int newColor,int firstColor){
        if(r<0 || r>=image.length || c<0 || c>=image[r].length){
            return;
        }
        if(image[r][c]==firstColor){
        image[r][c]=newColor;
        dfs(image,r-1,c,newColor,firstColor);
        dfs(image,r,c-1,newColor,firstColor);
        dfs(image,r+1,c,newColor,firstColor);
        dfs(image,r,c+1,newColor,firstColor);
        }
    }
}
```

Custom Testcase (Contribute)

Run Code Submit