Flood Fill LeetCode

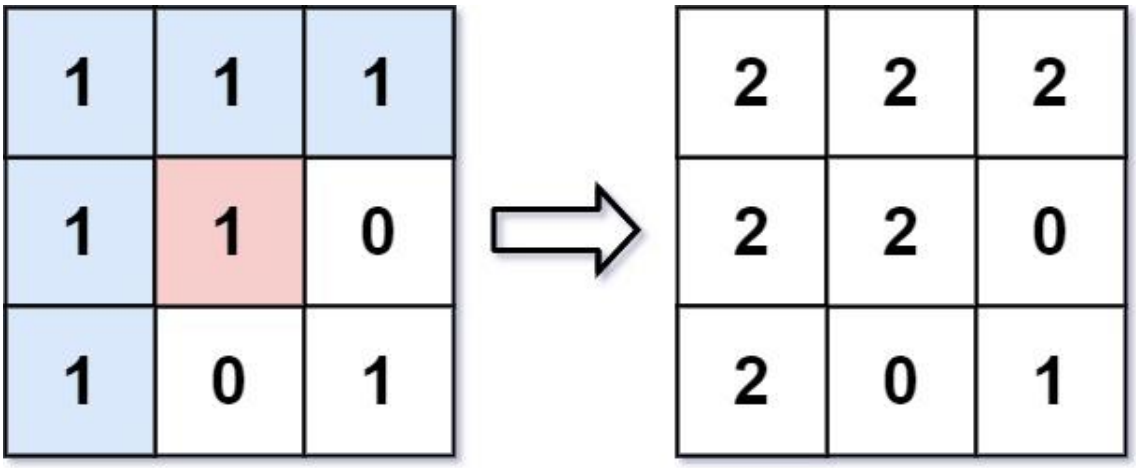
An image is represented by an m x n integer grid image where image[i][j] represents the pixel value of the image.

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:



Approach 1: Function to perform flood-fill using DFS

- **Explanation:**
    - The **floodFillDFS** function uses recursive DFS traversal to modify pixels in the image.
    - It explores neighboring pixels, checks conditions, and continues the traversal.
- **Time Complexity:**
    - **The time complexity is O(N * M), where N is the number of rows and M is the number of columns in the image.**
        - **Visiting each pixel once in the DFS traversal.**

- **Space Complexity:**
    - **The space complexity is O(N * M), attributed to the recursive call stack.**
        - **Storing information about each pixel during traversal.**

Approach 2: Function for flood fill using BFS

- **Explanation:**
    - The **floodFillBFS** function uses BFS traversal to modify pixels in the image.
    - It explores neighboring pixels, checks conditions, and enqueues valid pixels for further exploration.
- **Time Complexity:**
    - **The time complexity is O(N * M), where N is the number of rows and M is the number of columns in the image.**
        - **Visiting each pixel once in the BFS traversal.**
- **Space Complexity:**
    - **The space complexity is O(N * M), attributed to the BFS queue.**
        - **Storing information about each pixel during traversal.**

Conclusion

Both DFS and BFS approaches successfully achieve flood fill in the image. The time and space complexities for both approaches are O(N * M), where N is the number of rows and M is the number of columns in the image. The choice between them depends on specific requirements, such as memory constraints or desired exploration order. In this scenario, **both approaches demonstrate similar time and space complexities.**