# **Leetcode 542 – 01 Matrix**

## Problem Understanding

You are given an m x n binary matrix mat where:

* 0 represents a cell with a 0
* 1 represents a cell with a 1

**Return a matrix of the same size** where each cell with 1 contains the **distance to the nearest 0** (in terms of 4-directional moves).

### Example:

**Input:**

mat = [

[0,0,0],

[0,1,0],

[1,1,1]

]

**Output:**

[

[0,0,0],

[0,1,0],

[1,2,1]

]

## Optimized Java Solution (Multi-Source BFS)

class Solution {

public int[][] updateMatrix(int[][] mat) {

int rows = mat.length, cols = mat[0].length;

Queue<int[]> queue = new LinkedList<>();

boolean[][] visited = new boolean[rows][cols];

// Add all 0s to queue and mark them visited

for (int i = 0; i < rows; i++) {

for (int j = 0; j < cols; j++) {

if (mat[i][j] == 0) {

queue.offer(new int[]{i, j});

visited[i][j] = true;

}

}

}

int[][] dirs = {{0,1}, {1,0}, {0,-1}, {-1,0}};

// BFS from all 0s

while (!queue.isEmpty()) {

int[] curr = queue.poll();

int r = curr[0], c = curr[1];

for (int[] dir : dirs) {

int nr = r + dir[0];

int nc = c + dir[1];

if (nr >= 0 && nr < rows && nc >= 0 && nc < cols && !visited[nr][nc]) {

mat[nr][nc] = mat[r][c] + 1;

queue.offer(new int[]{nr, nc});

visited[nr][nc] = true;

}

}

}

return mat;

}

}

## Dry Run Using Table

### Input:

[

[0,0,0],

[0,1,0],

[1,1,1]

]

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Queue Front | Update | Grid Snapshot |
| Init | All 0s | - | 0 0 00 1 01 1 1 |
| 1 | (1,1) | → 1 | 0 0 00 1 01 1 1 |
| 2 | (2,0) | ← from (1,0) | 0 0 00 1 01 1 1 |
| 3 | (2,1) | ← from (2,0) or (1,1) | 0 0 00 1 01 2 1 |

**Final Output:**

[

[0,0,0],

[0,1,0],

[1,2,1]

]

## Time / Space Complexity

|  |  |
| --- | --- |
| Metric | Value |
| Time | O(m × n) |
| Space | O(m × n) |

* Each cell is visited once.
* BFS queue holds at most all cells.

## Alternate Approaches

|  |  |  |  |
| --- | --- | --- | --- |
| Approach | Time | Space | Notes |
| ✅ Multi-source BFS | O(m×n) | O(m×n) | Optimal |
| ❌ DFS from each 1 | O((m×n)²) | O(m×n) | Too slow |
| 🔍 DP in 2 passes | O(m×n) | O(1) extra | Requires 2 sweeps, clever math |