

Rotten Oranges

● MEDIUM

■ Introduction

Every minute, fresh oranges adjacent to rotten ones become rotten. Find the minimum time to rot all oranges, or return -1 if impossible. This is a classic **Multi-Source BFS** problem — multiple sources spread simultaneously.

■ Core Intuition

Multi-Source BFS

- Start BFS from ALL rotten oranges simultaneously (add all to queue at time 0).
- Each BFS level = 1 minute of rotting. Level count = time taken.
- After BFS, check if any fresh orange remains → return -1 if yes.
- Key: multi-source BFS ensures simultaneous spreading from all initial rotten oranges.

■ Visual

Rotten Oranges Trace

Initial: Min 1: Min 2:

2 1 1 2 2 1 2 2 2

1 1 0 → 2 1 0 → 2 2 0

0 1 1 0 1 1 0 2 2

Queue at start: [(0,0), (1,0)] – both rotten oranges

BFS level 1: rot (0,1),(1,1) → level 2: rot (0,2),(2,1),(2,2)

Answer: 2 minutes

■ Java Implementation

Java

```
class Solution {
    public int orangesRotting(int[][] grid) {
        int rows = grid.length, cols = grid[0].length;
        Queue<int[]> queue = new LinkedList<>();
        int fresh = 0;

        // Step 1: Add all rotten oranges to queue, count fresh
        for (int r=0; r<rows; r++)
```

```

for (int c=0; c<cols; c++) {
    if (grid[r][c] == 2) queue.add(new int[]{r,c});
    if (grid[r][c] == 1) fresh++;
}

if (fresh == 0) return 0; // No fresh oranges

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

// Step 2: BFS – spread rotting level by level
while (!queue.isEmpty() && fresh > 0) {
    minutes++;

    int size = queue.size();
    for (int i=0; i<size; i++) {
        int[] cell = queue.poll();
        for (int[] d : dirs) {
            int nr = cell[0]+d[0], nc = cell[1]+d[1];

            if (nr>=0 && nr<rows && nc>=0 && nc<cols && grid[nr][nc]==1) {
                grid[nr][nc] = 2;
                queue.add(new int[]{nr,nc});
                fresh--;
            }
        }
    }
}

return fresh == 0 ? minutes : -1;
}

```

Time & Space Complexity

Case	Time	Space	Why
Time	$O(m*n)$	$O(m*n)$	Every cell enqueued at most once
Space	$O(m*n)$	$O(m*n)$	Queue can hold all cells in worst case

Common Mistakes

Avoid These

- Single-source BFS — must start from ALL rotten oranges simultaneously.
- Not counting minutes correctly — increment once per BFS level, not per cell.
- Forgetting to check if fresh > 0 after BFS — isolated fresh oranges cant be reached.
- Modifying grid without tracking freshCount — count fresh separately.

■ Practice Problems

Case	Time	Space	Why
Medium	LC 994	Rotten Oranges	Direct application
Medium	LC 542	01 Matrix	Multi-source BFS from 0s
Medium	LC 1162	Far from Land	Multi-source BFS from land
Hard	LC 317	Shortest Distance from All Buildings	Multi-source BFS

■ Mental Model

Remember Forever

- "Multi-source BFS = multiple epidemics starting at time 0. Use BFS levels to count time. After BFS, isolated survivors = -1."