994. Rotting Oranges

Medium 3773 223 Add to List Share

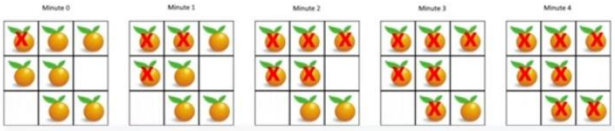
You are given an `m x n grid` where each cell can have one of three values:

- `0` representing an empty cell,
- `1` representing a fresh orange, or
- `2` representing a rotten orange.

Every minute, any fresh orange that is **4-directionally adjacent** to a rotten orange becomes rotten.

Return *the minimum number of minutes that must elapse until no cell has a fresh orange.* If *this is impossible, return* `-1`.

Example 1:




```
Input: grid = [[2,1,1],[1,1,0],[0,1,1]]
Output: 4
```

Example 2:

994. Rotting Oranges

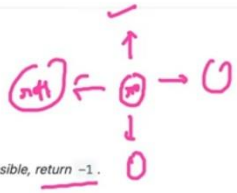
Medium 3773 223 Add to List Share

You are given an `m x n grid` where each cell can have one of three values:

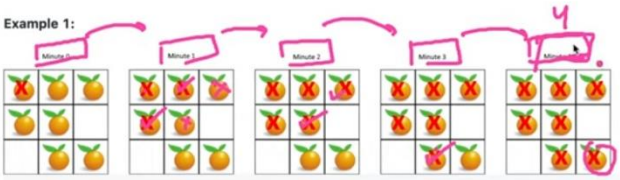
- `0` representing an empty cell,
- `1` representing a fresh orange, or
- `2` representing a rotten orange.

Every minute, any fresh orange that is **4-directionally adjacent** to a rotten orange becomes rotten.

Return *the minimum number of minutes that must elapse until no cell has a fresh orange.* If *this is impossible, return* `-1`.



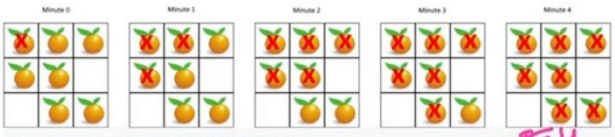
Example 1:




```
Input: grid = [[2,1,1],[1,1,0],[0,1,1]]
Output: 4
```

Example 2:

Example 1:



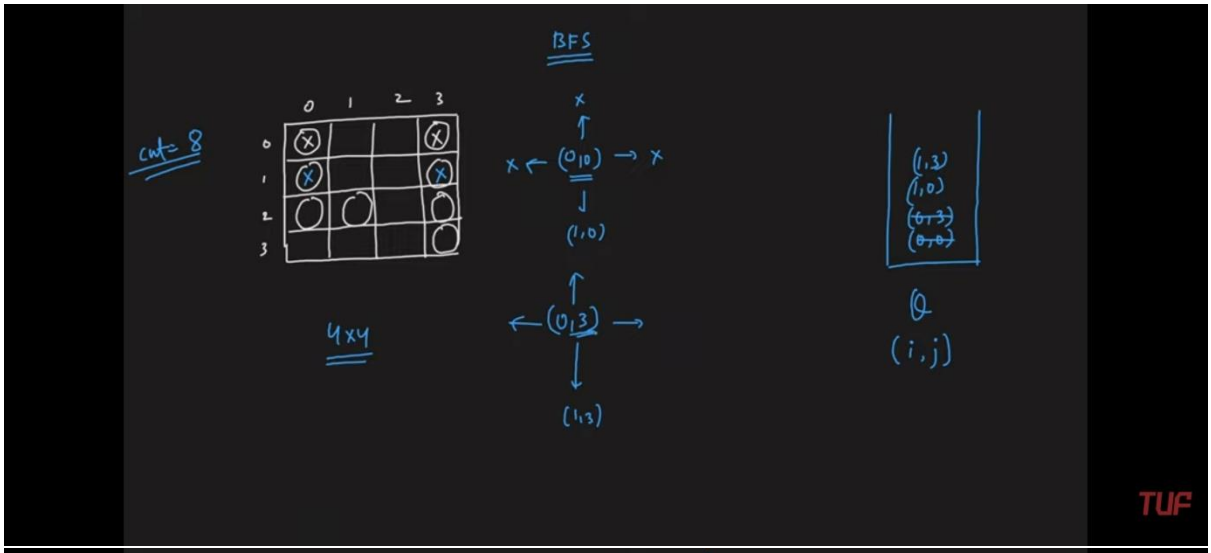
```
Input: grid = [[2,1,1],[1,1,0],[0,1,1]]
Output: 4
```

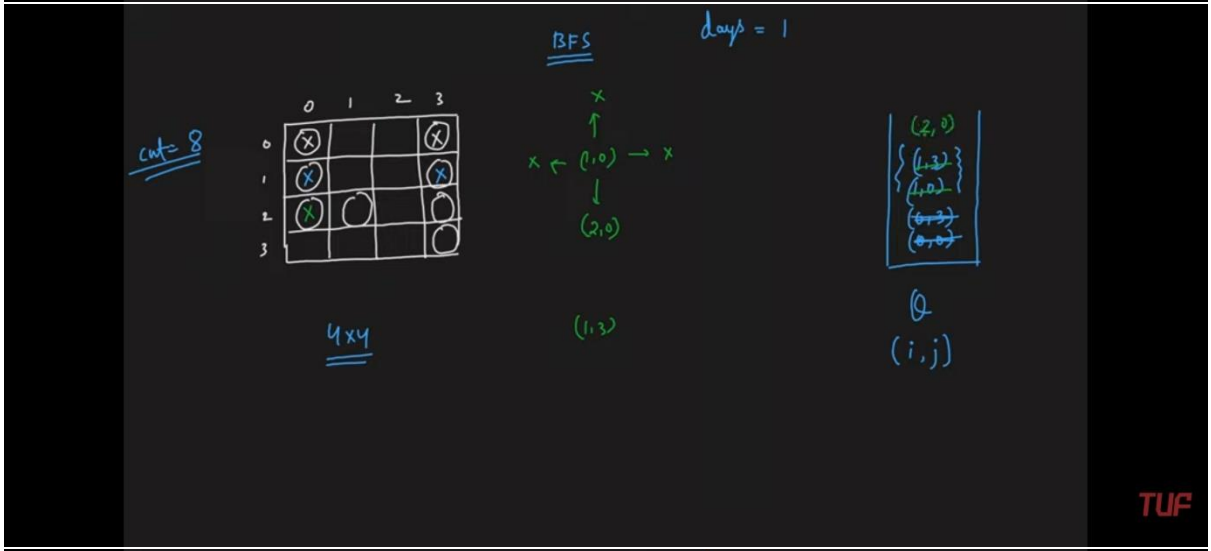
Example 2:

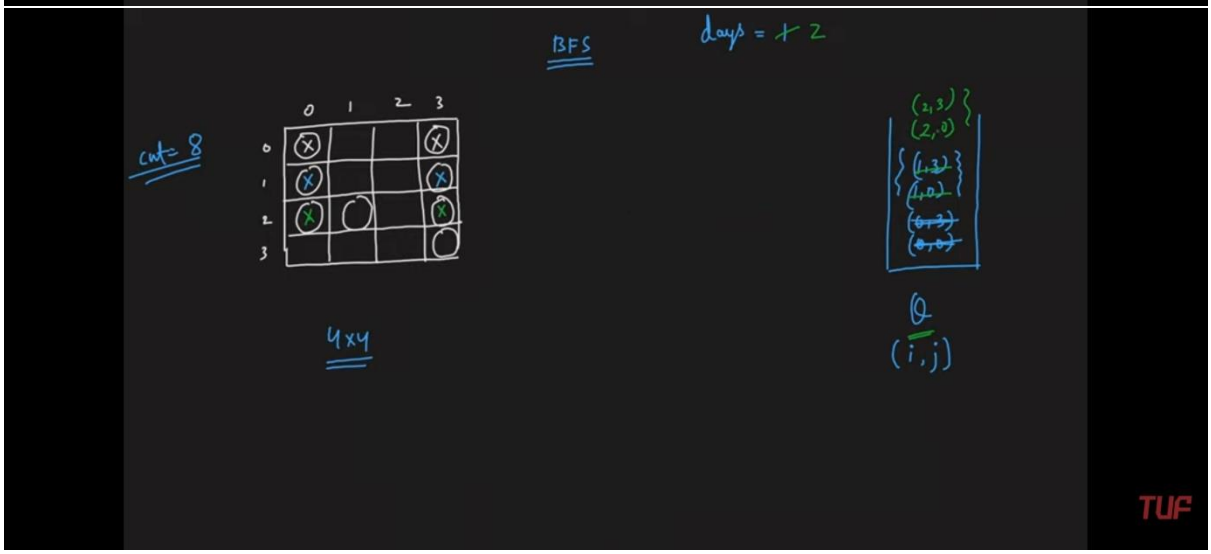
```
Input: grid = [[2,1,1],[0,1,1],[1,0,1]]
Output: -1
Explanation: The orange in the bottom left corner (row 2, column 0) is never rotten, because rotting only happens
4-directionally.
```

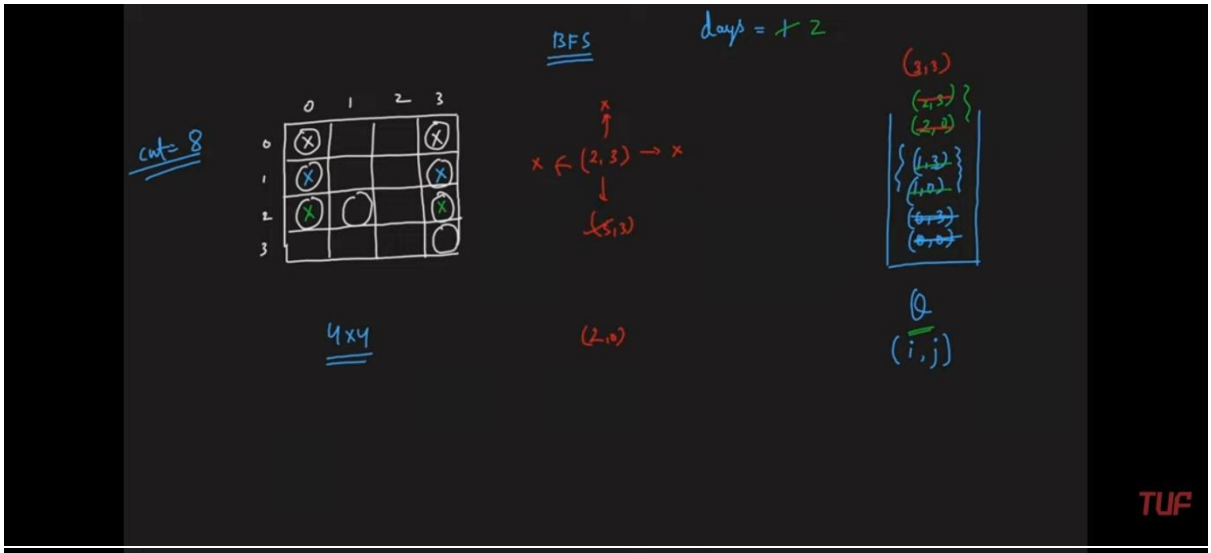
Example 3:

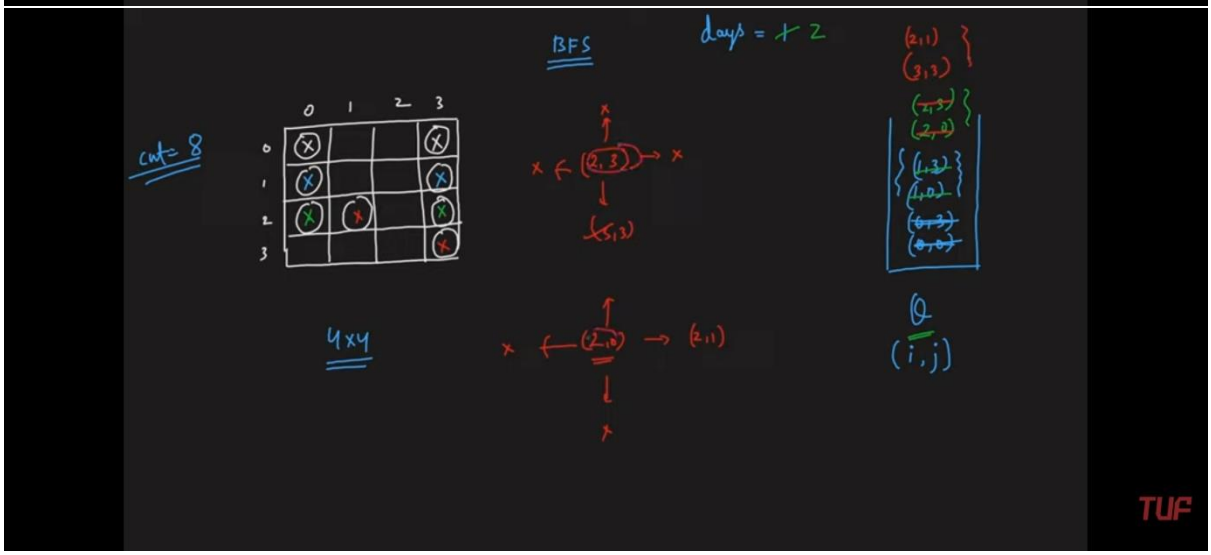
```
Input: grid = [[0,2]]
Output: 0
Explanation: Since there are already no fresh oranges at minute 0, the answer is just 0.
```

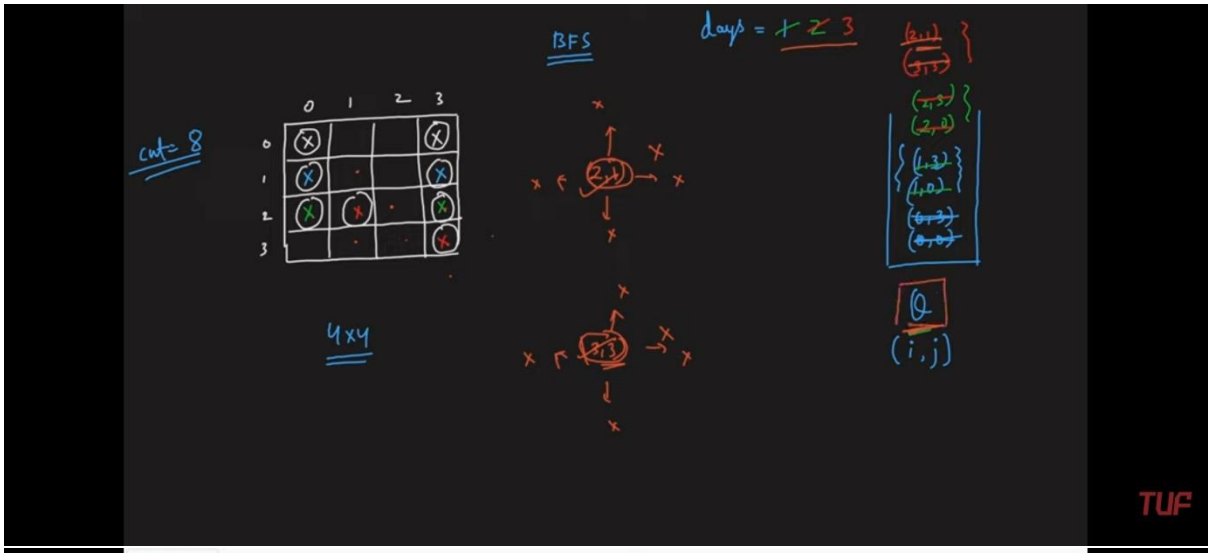

BFS
cnt = 8
0 1 2 3
0
1
2
3
4x4
(0,0)
(0,3)
(1,0)
(1,3)
Q
(i,j)
TUF


days = 1
BFS
cnt = 8
0 1 2 3
0
1
2
3
4x4
(2,0)
(1,3)
Q
(i,j)
TUF


days = + 2
BFS
cnt = 8
0 1 2 3
0
1
2
3
4x4
(2,3)
(2,0)
(1,3)
(1,0)
(0,3)
(0,0)
Q
(i,j)
TUF


BFS
days = + 2
cnt = 8
0 1 2 3
4x4
(2,3)
(2,0)
(3,3)
(i,j)
TUF


BFS
days = + 2
cnt = 8
0 1 2 3
4x4
(2,3)
(2,0)
(2,1)
(3,3)
(i,j)
TUF



C++ • Autocomplete

```cpp
class Solution {
public:
    int orangesRotting(vector<vector<int>>& grid) {
        if(grid.empty()) return 0;
        int m = grid.size(), n = grid[0].size(), days = 0, tot = 0, cnt = 0;
        queue<pair<int, int>> rotten;
        for(int i = 0; i < m; ++i){
            for(int j = 0; j < n; ++j){
                if(grid[i][j] != 0) tot++;
                if(grid[i][j] == 2) rotten.push({i, j});
            }
        }

        int dx[4] = {0, 0, 1, -1};
        int dy[4] = {1, -1, 0, 0};

        while(!rotten.empty()){
            int k = rotten.size();
            cnt += k;
            while(k--){
                int x = rotten.front().first, y = rotten.front().second;
                rotten.pop();
                for(int i = 0; i < 4; ++i){
                    int nx = x + dx[i], ny = y + dy[i];
                    if(nx < 0 || ny < 0 || nx >= m || ny >= n || grid[nx][ny] != 1) continue;
                    grid[nx][ny] = 2;
                    rotten.push({nx, ny});
                }
            }
            if(!rotten.empty()) days++;
        }

        return tot == cnt ? days : -1;
    }
};
```