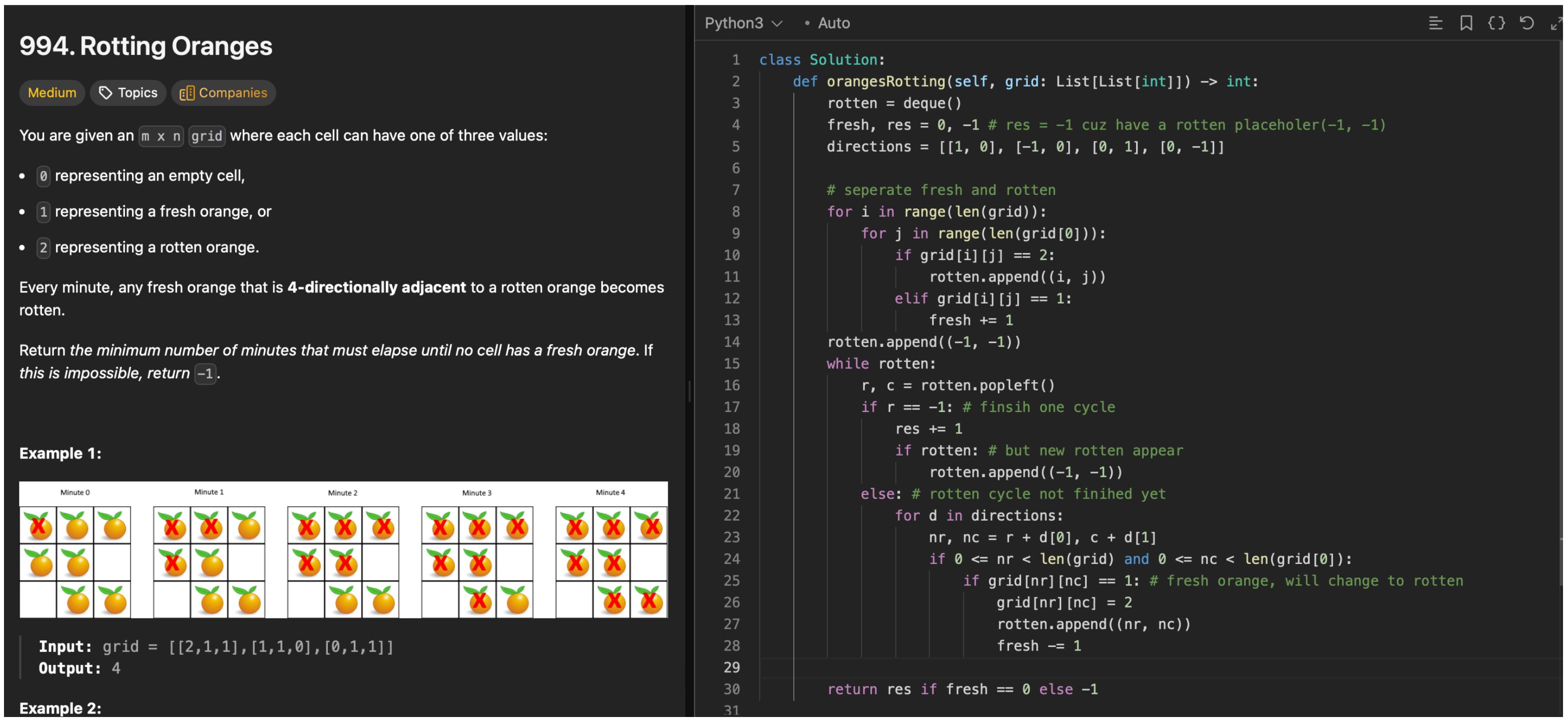


994. Rotting Oranges

Medium · Topics · Companies

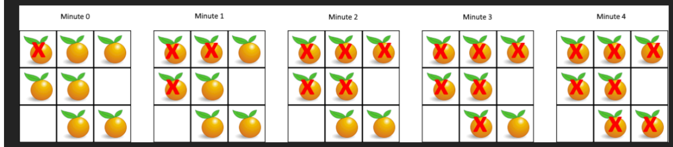
You are given an `m x n` `grid` where each cell can have one of three values:

- `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:**

Python3 · Auto

```python
class Solution:
    def orangesRotting(self, grid: List[List[int]]) -> int:
        rotten = deque()
        fresh, res = 0, -1 # res = -1 cuz have a rotten placeholer(-1, -1)
        directions = [[1, 0], [-1, 0], [0, 1], [0, -1]]

        # seperate fresh and rotten
        for i in range(len(grid)):
            for j in range(len(grid[0])):
                if grid[i][j] == 2:
                    rotten.append((i, j))
                elif grid[i][j] == 1:
                    fresh += 1
        rotten.append((-1, -1))
        while rotten:
            r, c = rotten.popleft()
            if r == -1: # finsih one cycle
                res += 1
                if rotten: # but new rotten appear
                    rotten.append((-1, -1))
            else: # rotten cycle not finihed yet
                for d in directions:
                    nr, nc = r + d[0], c + d[1]
                    if 0 <= nr < len(grid) and 0 <= nc < len(grid[0]):
                        if grid[nr][nc] == 1: # fresh orange, will change to rotten
                            grid[nr][nc] = 2
                            rotten.append((nr, nc))
                            fresh -= 1

        return res if fresh == 0 else -1
```

bfs with queue

unlike normal bfs, append(-1, -1) each time when a cycle of precess finish, that's why need to set res to be -1 in the first place